Generating Personalised Service Recommendations

by

Gregory Huczynski

A thesis submitted to the
Department of Computing Science
at the University of Glasgow
for the degree of
Doctor of Philosophy

October 2004

Abstract

In the context of service-oriented computing, the issue of service selection is an important one: how can a consumer find and choose a single, appropriate service of the required type, given the mass of services potentially available on a network? By using a service discovery mechanism (the focus of current service selection research), a consumer is able to obtain an unordered list of services which match explicitly specified requirements, from which he must select the service he considers most appropriate. However, formulating the original service request and selecting a service from the returned list are both challenging tasks, particularly for a consumer in unknown circumstances, with unknown services available.

This research is thus concerned with the investigation, development and evaluation of a general design for a system that can provide a *personalised service recommendation* of appropriate services to a requesting consumer. The personalised service recommendation is generated through the assessment of past service selections/usage. A design-adhering prototype has been demonstrated to generate effective personalised service recommendations in a real-world scenario.

Acknowledgements

I should like to express my gratitude to my supervisors, Peter Dickman and Phil Gray, for their support and helpful advice throughout the period of my research. I should also like to thank Gary Gray and other members of the Systems Development Group for their technical assistance. In addition, John McColl of the Department of Statistics at Glasgow University provided useful statistical advice in connection with evaluation, and my thanks go to him. Gratitude is also due to Glasgow University, for allowing me to use the data on which my research is based, and the Carnegie Trust, who generously funded me.

On a more personal note, I am grateful to my office-mates, Olufemi Komolafe and Steven Heeps, for providing support and for proof-reading my thesis. I should also like to thank, in alphabetical order, Barry Brown, Michael Dales, Iain Darroch, Huw Evans, Areti Galani, Matt Holgate, Gareth P. McSorley, Rolf Neugebauer, Jonathan Paisley, Rebecca Randell and Lyndell St. Ville, who all helped to make my PhD experience an enjoyable one.

Last, but not least, thanks go to Mum, Dad, Sophie and Tom. In particular, I wholeheartedly thank my mother for her constant support, encouragement, proof-reading and comfort food. Mum, you kept me sane.

Contents

List of Figures

Chapter 1

Introduction

1.1 Background to Research

Interest in the concept of service-oriented computing is growing rapidly, not only in academia but also in industry. Generally speaking, a service can be defined as a network-accessible, self-contained software component that provides a particular type of functionality through a well-defined interface. With service-oriented computing, applications will be partially or completely composed of services available on an intranet or the Internet.

Allied to the concept of service-oriented computing is the issue of service selection. There may well be a myriad of services available on a network, but how can a single, appropriate service of the required type be found and chosen? How can a consumer, i.e. a user or his proxy application, select the best service for his specific circumstances?

Current research into service selection has focused on the development of service discovery mechanisms (SDMs), which essentially take the form of automated "Yellow Pages". At the core of an SDM system is some form of service "registry", which acts as a third-party broker between services and consumers in a network. In order to locate a service, a consumer must submit a request to the registry, specifying the service type and any other associated requirements. The registry compares this request against advertised service descriptions, and returns the set of available services which match the requirements. The consumer must then choose a service from this unordered set. Thus, the onus is on the consumer, both to formulate a precisely defined service request and then to decide which of the services in the returned, possibly large, set is most appropriate. These are challenging tasks, particularly for a consumer in unfamiliar circumstances where unknown services are available.

For such an uninformed consumer, a style of SDM registry which actually recommends appropriate services of the required type - a *recommending registry* - would surely be of distinct benefit. In response to a service request, the registry itself would determine the appropriateness for the requesting consumer of available type-matching services. It would then order them by their perceived appropriateness, ranking them from most to least appropriate. Finally, the recommending registry would return this ordered list to the requesting consumer as a *personalised service recommendation.* If the highly ranked services were truly appropriate choices, enabling the consumer to search through the recommendation and select an ideal service with minimum time and effort, then the recommendation could be considered effective.

1.2 Aim of Research

Although some interest has been shown in the concept itself, no-one has yet comprehensively explored the issue of registry-generated personalised service recommendation with a view to finding a general solution to the general problem of service selection. For the problem *is* a general one: regardless of service type, regardless of scenario, the difficulties associated with finding and choosing an appropriate service from the mass of services available are universal. In view of this, my own research has focused on the problem of generality, with the precise aim of addressing the following question:

> What general design for an SDM recommending registry would enable the generation of effective personalised service recommendations?

Theoretically, a general design would be of considerable value because it could provide a blueprint for the construction and operation of any SDM recommending registry, regardless of deployment scenario or service types involved. Hopefully, the design that I have devised fulfils this criterion.

1.3 Research Undertaken

After some initial research into the problem, I concluded that, although theoretically feasible, a general design derived from current service selection techniques would have several significant drawbacks. In consequence, I devised my own, novel approach to registry-generated personalised service recommendation, based on the assessment of past service

selections/usage. This approach can be seen as a form of Collaborative Filtering.

In order to develop my approach into a general design for a recommending registry, I first identified relevant research issues. I then proceeded to devise basic solutions to these issues and formulated a basic design from them. This design was assessed for viability and validity through the construction and evaluation of a working, design-adhering prototype recommending registry in a real-world scenario. In order to evaluate the prototype registry for effectiveness, in terms of the recommendations that it generated, I created an original evaluation scheme, taking my initial inspiration from Information Retrieval. The basic design was shown to be both viable and valid and was therefore considered worthy of further investigation and development.

Through assessment of the basic design, in terms of the evaluation results and the original research issues, I identified three elements of the design which would benefit from further improvement. The core problem associated with each of these elements was then pinpointed, and I subsequently investigated the problems and devised successful solutions to them. Ultimately, I formulated a more advanced design for a recommending registry incorporating the three improvements made. It is this design that I am presenting as a general design for an SDM recommending registry that would enable the generation of effective personalised service recommendations.

As my approach to registry-generated personalised service recommendation is a novel one, the research undertaken should be seen as an exploratory attempt to determine whether such an approach can form the basis of a viable and valid general design for a recommending registry. No claim is made that the resultant design is definitive, only that it is a solution that vindicates the assertion made in the thesis statement below (Section 1.4).

1.4 Thesis Statement

The thesis statement is as follows:

> I assert that it is possible to devise a general design for an SDM recommending registry which would enable the generation of effective personalised service recommendations based on an assessment of past service selections/usage. I shall demonstrate the validity of this assertion by developing such a design, constructing a design-adhering working prototype recommending registry, and

evaluating it for effectiveness, within a real-world scenario.

1.5 Research Contributions

The two main contributions of this research are:

- A general design for an SDM recommending registry.
- An evaluation scheme which can be used to assess the effectiveness of any recommending registry that adheres to this general design.

There are also two other contributions, which are detailed in the Conclusion (Chapter 14).

1.6 Thesis Structure

The remainder of this thesis is structured as follows:

Chapter 2: The Concept of Service-Oriented Computing This chapter contains information relating to the concept of service-oriented computing, together with details of three application areas.

Chapter 3: The Issue of Service Selection This chapter contains information about current service selection research, including details and analysis of various SDM systems.

Chapter 4: Towards Personalised Service Recommendation This chapter focuses on the issue of advancing from service discovery to service recommendation, with details given of an abstract model for a proposed SDM recommending registry, together with a discussion and analysis of existing relevant research. The aim of my research is also defined here.

Chapter 5: A Theoretical General Design for a Recommending Registry This chapter contains details of a theoretical general design for a recommending registry, derived from current service selection techniques. An assessment of the design is then given.

Chapter 6: A Novel General Approach Based on Collaborative Filtering This chapter focuses on my novel approach to registry-generated personalised service recommendation, based on the assessment of past service selections/usage. The approach is defined

and justified, the related research issues discussed, and the association of the approach with Collaborative Filtering noted.

Chapter 7: Evaluating Recommending Registry Effectiveness This chapter contains a detailed explanation and definition of my novel evaluation scheme for assessing recommending registry effectiveness. This is the evaluation scheme specified in the Research Contributions (Section 1.5).

Chapter 8: A Basic Recommending Registry Design This chapter contains details of the basic recommending registry design, including information concerning the basic solutions to the research issues.

Chapter 9: Assessing the Basic Recommending Registry Design This chapter contains information about the assessment of the basic design through the construction and evaluation of a design-adhering working prototype recommending registry in a real-world scenario.

Chapter 10: An Advanced Recommendation Generation Algorithm This chapter contains details relating to the first improvement made to the basic design, namely the development of an advanced recommendation generation algorithm.

Chapter 11: Relaxing the Test for Service Selection Situation-Similarity This chapter contains information concerning the second improvement made to the basic design, namely the relaxation of the test for service selection situation-similarity.

Chapter 12: Recommending Registry Configuration using Self-Optimisation This chapter contains details relating to the third and final improvement made to the basic design, namely the simplification of the developer task of registry configuration.

Chapter 13: An Advanced Recommending Registry Design This chapter provides a definition of the advanced recommending registry design, which incorporates the improvements detailed in the last three chapters. This is the general design specified in the Research Contributions (Section 1.5).

Chapter 14: Conclusion This chapter contains a discussion of the completed work and suggestions for future investigation. My research contributions are also detailed here.

Appendix A: Glossary of Acronyms This appendix provides a glossary of important acronyms used throughout this thesis.

Appendix B: Chapter 9 Details This appendix contains information relating to Chapter 9.

Appendix C: Chapter 10 Details This appendix contains information relating to Chapter 10.

Appendix D: Chapter 11 Details This appendix contains information relating to Chapter 11.

NB: The pronoun "he" has been used throughout this thesis to denote, what the Oxford Dictionary (1990) defines as, "a person, etc. of unspecified sex". Obviously, if appropriate, such a "he" can also mean "she".

Chapter 2

The Concept of Service-Oriented Computing

The concept of service-oriented computing is presented in this chapter. A service is defined, the features and benefits of service-oriented architecture are discussed, and the current interest and activity in the area highlighted. Finally, three application areas where service-oriented architecture is considered of increasing relevance are identified and detailed.

2.1 The Concept of Service-Oriented Computing

With the explosive growth of networking in the last decade, there has been burgeoning interest in the concept of service-oriented computing [89,90]. Generally speaking, a service can be defined as a network-accessible, self-contained software component that provides functionality through a well-defined interface. It could provide any form of functionality: for example, the use of a physical device such as a printer or projector, or that of a purely software resource such as a stock-ticker or search-engine. Proponents of service-oriented architecture (SOA) [8,77] envisage applications which are partially or completely composed of services available on an intranet or the Internet.

Theoretically, SOA could provide several benefits, which are derived from two key features of the architecture. Firstly, the functionality of a service is exposed via an interface, which is a contract for the service's semantics that defines how the service will behave and the syntax of interactions with the service. An application is built to interact with an interface which provides required functionality, rather than with a specific service

which supports that interface. Secondly, an application dynamically finds and binds to the particular services that it utilises at runtime.

From a software-engineering perspective, SOA has many of the benefits of the object-oriented and component-based programming paradigms, but on a distributed scale. The "black-box" nature of a service hides complexity, is amenable to unit testing, and encourages reuse of services in multiple applications. By linking together the functionality provided by different services, it should be possible to build complex distributed applications rapidly. The combination of interface interaction and runtime service binding could provide flexibility and fault-tolerance. When an application is running, it may have a choice between several deployed services which support a particular interface that it requires, and could dynamically bind to the service that appears most appropriate, such as the one that seems least loaded. Moreover, if this service fails or is shutdown for maintenance purposes, the application could continue operating normally by rebinding to an alternative service.

2.2 The Current State of Service-Oriented Computing

Service-oriented computing is not a new concept. Since the early 1990s, various distributed middleware solutions have been released, such as CORBA [47], Java RMI [84], Jini [37] and .NET [82], which enable systems to be built that conform to many of the principles of SOA. Each middleware solution provides a different method of interface definition, service implementation and service interaction, which are all incompatible with one another. Recently, however, there has been a sudden upsurge of interest in SOA, with a number of articles [28, 52, 103–105] and industry white-papers [61, 110] being published on the World Wide Web which extol the benefits of the concept. As noted by McGovern et al [77], this upsurge appears to have been caused by the recent development of Web Service standards.

Web Service standards [22] are an industry-wide effort to enable the construction of interoperable SOA-style services. In much the same way that the success of the World Wide Web was driven by the proposal and adoption of simple, open Internet standards such as HTTP (Hyper-Text Transfer Protocol) and HTML (Hyper-Text Markup Language), similar attempts are being made to drive the growth of service-oriented computing through the proposal of open standards that define various aspects of SOA. Two core XML-based (eXtensible Markup Language [20]) standards have been proposed: WSDL (Web Services

Description Language) [23] and SOAP (Simple Object Access Protocol) [21]. WSDL can be used to define service interfaces in a platform-independent manner, whilst SOAP provides a platform-independent message format for service interaction.

Web Service standards focus on interoperability. Unlike many of the existing heavyweight middleware solutions, the standards do not define any aspects of service implementation. The expectation is that an application written in one programming language running on one platform will be able to utilise a service written in a different programming language running on a different platform, assuming that all interaction between the two conforms to the platform-independent Web Service standards. For example, a .NET application running on Windows XP could utilise a Java service running on Linux, assuming that all interaction was phrased in terms of operations defined in the WSDL service interface, with component messages being encoded as SOAP running over an Internet transport protocol such as HTTP.

Activity in the area of service-oriented computing is increasing, in both industry and academia. In industry, major software companies such as Microsoft, Sun and IBM have started to provide development tools and application frameworks with Web Services and SOA support. Indeed, Microsoft has announced that an integral element of its next-generation Longhorn operating system will be Indigo [81], a unified programming model and communications infrastructure based on SOA principles. In academia, the main topic of the Communications of the ACM October 2003 issue [90] was service-oriented computing, and the first conference [60] dedicated to the concept was held in December 2003.

2.3 Applications of Service-Oriented Computing

Much of the recent journalistic coverage of service-oriented computing has been phrased from the perspective of the business enterprise. Industry analysts are promoting SOA as an ideal means of structuring software within the enterprise, emphasising, in business terms [28, 61], many of the benefits detailed earlier in Section 2.1. They assert that by structuring business logic as a suite of reusable services, application development time and costs can be reduced, and return on investment maximised. Moreover, they argue that SOA can increase "business intelligence", by enabling information to be shared between existing monolithic applications that did not previously communicate with one another. For example, legacy systems could be integrated into developed enterprise applications,

together with newly-built services, by wrapping them up as services themselves. Indeed, a legacy service could eventually be replaced in the enterprise with minimal upheaval by deploying a new service which supported the same interface.

However, SOA has much greater potential than merely being the latest fashionable architecture for structuring software within the enterprise. SOA can also be considered a suitable enabling technology for several new and expanding areas of computing: the "services web", ubiquitous computing and grid computing. These three areas are discussed below.

2.3.1 The "Services Web"

Currently, most deployment of services appears to be occurring in the business world. It seems likely that enterprises will initially use SOA to structure internal software within their intranets. However, there is an expectation that, over time, enterprises will begin making services available on the Internet for other parties to take advantage of [33, 61]. For example, an enterprise in a supply chain might offer an inventory information service for other partners in the chain to query. Alternatively, a shopping web-site might offer an item purchasing service, which could be integrated into a customer's personalised desktop shopping application.

Indeed, some experts envisage a market of services [89,90], in which enterprises would compete to provide a particular type of service to others; service interfaces would need to be standardised, so that providers could develop competing implementations of a particular functionality. For example, the item purchasing system of the shopping web-site might require credit-card validation. a service offered by multiple providers. An inexpensive service might initially be chosen for use in the application. If that service proved unsatisfactory, perhaps by being unreliable, the running application could be dynamically switched to using an alternative service offered by another provider.

Gradually, services could be made available on the Internet by providers other than business enterprises. For example, local and national government organisations, educational institutions, health-care providers, and even individuals, could all deploy services which they considered of use to others. In essence, a "World Wide Web" of services would be created, but a web of dynamic functionality, not of static data. This services web could provide significant benefits and opportunities. Business-to-business collaboration could be streamlined, with the dynamic integration of different enterprises' services, and new

business models created. Novel and useful applications could be developed, composed of disparate services offered by different providers. Moreover, the potential selection of services available that provided a particular functionality would enable the luxury of choice, with the most appropriate service being selected for use in an application.

2.3.2 Ubiquitous Computing

Like the services web, ubiquitous computing could also have a significant impact on society, but from a somewhat different perspective. First espoused by Mark Weiser over a decade ago [118], the concept generally refers to the notion of "computing everywhere"; rather than being constrained to the desktop in the workplace or home, computing would be seamlessly integrated into the physical world. Physical objects would be embedded with networked computing capability, augmenting their functionality or enabling their remote control [10]. Moreover, purely software resources would be associated with places and their corresponding activities. In such a world, computing could begin to fulfil a significant role in supporting people's everyday lifestyles. As people went about their daily lives, they could access relevant computing resources to aid them in their activities or to augment their real-world experiences.

Recent technological advances and trends should enable the vision of ubiquitous computing to become a reality in the near future. The growth of smart-phone and laptop usage, allied with the widespread deployment of wireless networks to support voice and data traffic, should provide a solid technological base for ubiquitous computing. Currently, someone equipped with a mobile computing device can access network-accessible computing resources through local wireless networks. In such an environment, "ubicomp" systems could begin to be deployed.

As a clarifying illustration, imagine an ubicomp world in which "Alice" visits a university department to give a presentation. On arriving in the university town, Alice chooses a local mapping application using her smart-phone, and obtains directions to the relevant university building. Having arrived at the department, Alice prepares for the presentation by setting up her wireless-enabled laptop in the meeting room. The presentation application accesses the meeting-room lights and digital projector, enabling Alice to configure and control them remotely. After the presentation, an audience member asks for a copy of the slides, and Alice obliges by printing a paper version to one of the nearby printers. As a thank-you, Alice is taken out to dinner by a departmental research group. In order not

to be late for the train which takes her home, Alice decides to pre-book a taxi to take her to the station later in the evening. She uses her smart-phone to browse through several taxi services, and selects one to collect her from the restaurant at a specified time.

Although the technology may be mature enough to enable the example of Alice to be feasible, there is still no clear consensus as to what underlying middleware is required to facilitate ubiquitous computing. Various research middleware solutions have been constructed, such as Cooltown [65], iCrafter [95], Speakeasy [38] and MobiShare [114], which each provide different methods for packaging, finding and interacting with network-accessible ubicomp resources. However, there does appear to be a growing opinion [31,66] that SOA is an ideal means of structuring ubiquitous computing. Many of the research middleware solutions conform to SOA-style principles, and prototype ubicomp systems have been built using both Jini [16] and Web Services [25,76,114].

The core features of SOA fit well with the concepts of ubiquitous computing. Many aspects of the real world, such as objects and places, can logically be considered discrete elements associated with or providing particular functionality. This interpretation maps naturally onto the concept of services, which can provide computing manifestations of real-world elements. Thus, an SOA-structured ubicomp environment would consist of real-world associated services, which could be utilised by ubicomp applications over a wireless network.

Moreover, the dynamic nature of ubiquitous computing makes runtime binding a necessity [66,76]. Firstly, an ubicomp application is likely to be designed to utilise available services most appropriate to the user's current situation, which will vary over time. An application must use runtime binding in order to switch to using a more appropriate service when the user's situation changes, or when a better service becomes available. For example, Alice's presentation application would be designed to bind dynamically to projector and lighting services near to Alice's current location. Secondly, a particular service might not be available for the duration that its functionality is needed, requiring an application to rebind to an alternative if normal operation is to be maintained. As people go about their daily lives, they are likely to pass through multiple ubicomp environments controlled by different organisations [66], which may map onto different wireless networks. Certain services may only be available within a particular environment. If a person leaves an environment, alternatives may need to be found for those services which become unavailable. For example, when Alice leaves the university ubicomp environment on exiting

the departmental building, it is unlikely that she will be able to continue to use a printer service there. An alternative printer would need to be found in her current environment.

The benefits of SOA would seem to be of value to ubiquitous computing. Adoption of SOA would be another step closer to the realisation of an ubicomp world.

2.3.3 Grid Computing

Another area currently attracting significant attention in addition to ubiquitous computing is grid computing [40]. Originating in the scientific computing community, grid computing grew out of the realisation that more could be achieved through the pooling and sharing of scientific computing resources. By taking advantage of multiple parties' resources shared over the Internet, scientific tasks could be achieved which a single party, such as an individual, research group, or even research laboratory, could not achieve alone. For example, grid computing has been used to enable remote access to specialised experimental facilities such as earthquake simulators, and for distributed analysis of large amounts of data such as particle accelerator runs [41]. The term "grid computing" appears to have come about through the association of this scenario with that of a power grid. Like someone plugging an appliance into the power grid for it to operate, a scientific researcher would access a networked "grid" of scientific resources in order to "power" their experiment.

Initially, grid computing seemed to be mainly concerned with the pooling and sharing of raw computational and storage capacity of a large number of networked computers. More recently, however, grid computing appears to be moving towards a more service-oriented vision [41, 43]. The expectation is that a grid will consist of a market of scientific services offered by a range of different parties over the Internet, and that grid applications will be dynamically composed of these services. The services offered could provide varied functionality, not merely the use of computational and storage capacity. Service functionality could include the control of remote scientific equipment, access to databases, and usage of specialised experiment software.

As a clarifying example, imagine a grid application that has been assembled by "Bob", a bioinformatics researcher, to analyse a particular aspect of animal genomes. Bob has organised the application as an experiment workflow. Firstly, the application utilises a database-access service offered by a government-funded research institute to obtain the required genome data. The data is then partially processed by a commercial DNA sequence-analyser service, which, in Bob's opinion, provides a good trade-off between price and speed

compared to the alternatives. Finally the partial results are processed and visualised using software custom-built by Bob's research group.

To support this vision of grid computing, various middleware solutions have been built which conform to SOA principles, such as Globus [45] and Gridbus [13]. Moreover, a concerted effort is being made to standardise next-generation service-oriented grid computing through the definition of the Open Grid Services Architecture (OGSA) [42]. Building on Web Service standards, OGSA provides specifications for interoperable grid services and associated support systems through the definition of WSDL interfaces. If widely adopted, OGSA could enable the creation of a massive grid which pools and shares scientific computing resources on a worldwide scale.

2.4 Summary

In this chapter, both an introduction to and overview of service-oriented computing have been provided. The sudden upsurge of interest in SOA has been attributed to the recent development of interoperable Web Service standards, and a corresponding increase in SOA activity, both in industry and academia, noted. Business interest in SOA has been outlined, and the value of SOA as a suitable enabling technology within the areas of the services web, ubiquitous computing and grid computing identified. Finally, the role of SOA within these three areas has been discussed.

It is clear that service-oriented computing is of growing importance. However, numerous research problems will need to be overcome before the full potential of the concept can be realised. One of these, the issue of service selection, will be considered in the next chapter.

Chapter 3

The Issue of Service Selection

In this chapter, the issue of service selection is considered. More precisely, how is an appropriate service dynamically selected for use in an application? Service discovery mechanisms, which have been the focus of service selection research so far, are discussed, with a general description of how they operate being given. A more in-depth explanation is presented through the consideration of three representative example systems. Both an overview and an analysis of the core approach taken by service discovery research are then provided, and the conclusion is made that service discovery needs to change, better to support the uninformed consumer.

3.1 The Issue of Service Selection

Although the services web, ubiquitous computing and grid computing may differ in their core motivations, researchers have noted distinct similarity between the areas [42, 106] through their shared vision of a service-oriented future. All areas envisage parties offering services over a network, with applications dynamically composed of these services. All are therefore concerned with a number of common research issues associated with service-oriented computing. Of significant concern is the question of how an appropriate service is dynamically selected for use in an application. From the myriad of services available, how can a service be found which provides the required functionality and also proves to be an effective resource?

Current research in this area has been directed towards the development of service discovery mechanisms, which essentially act as a form of automated "Yellow Pages". The

aim of a service discovery mechanism (SDM) is to enable a service consumer (a user or his proxy application) to find an available service which matches specified requirements, to bind to it over the network, and then to interact with it. This sequence of events is commonly referred to as "find, bind and execute".

3.2 Service Discovery Mechanisms

Many of the SOA middleware solutions mentioned earlier, such as CORBA [47], Jini [37], Cooltown [65] and Globus [45], have inbuilt service discovery mechanisms. In addition, standalone service discovery mechanisms have also been built, such as SLP [49], Salutation [19], UPnP [83], UDDI [113], SSDS [27], INS [1] and Splendor [124]. These standalone systems are concerned purely with the "find and bind" aspects of service discovery, and are generally indifferent to service execution specifics. Various surveys and comparisons of different SDM systems have been undertaken [7, 53, 71, 123].

Despite the variety of SDM systems in existence, and the fact that they focus on numerous different technical aspects such as scalability, security and fault tolerance, all generally operate in much the same way, and conform to the same underlying structure. Figure 3.1 shows the core structure of an SDM system, and the sequence of events which comprise the service discovery process. At the core of an SDM system is some form of service "registry", which acts as a third-party broker between services and consumers in a network. When a service becomes available, a provider-defined description of the service is advertised in a network-accessible service registry (step 1). This description contains the service's network-address and type, which defines its functionality (perhaps as an interface signature). The description may also contain other information considered of relevance to the consumer in choosing a service. For example, the description might contain details of the service's capabilities, quality of service, price, and physical location (if the service corresponds to a real-world entity).

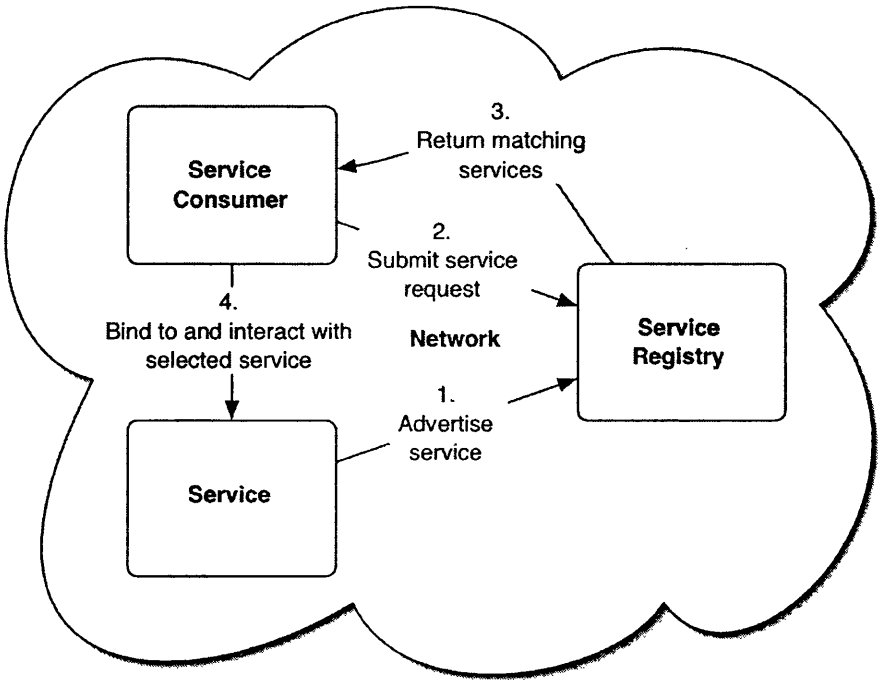


Figure 3.1: The Sequence of Events in Service Discovery

When a consumer requires a service, a request containing the service type and other requirements is submitted to the service registry (step 2). The registry compares this request against the advertised service descriptions, and returns the set of network-addresses of services which match the requirements (step 3). The consumer chooses a service from this unordered set, and binds to its network address (step 4). Interaction with the selected service can then occur.

It should be noted that a number of SDM systems [54, 102] have also been built that do not conform to the registry-based structure described above. Instead, these systems operate without a registry, with consumers generally broadcasting service requests onto a network and matching services responding. However, these systems will not be discussed further, as the majority of service discovery mechanisms do use a registry structure. Moreover, it is questionable how generally applicable broadcast-based SDM systems can be to service-oriented computing, since this style of system is not particularly scalable, and requires a broadcast medium such as a wireless connection or LAN multicast to operate.

Given the basic functional commonality of most registry-based SDM systems, a clear understanding of service discovery can be gained through the detailed consideration of only a small number of representative examples. Consequently, three frequently-cited SDM systems will be discussed below: SLP [49], the Jini Lookup Service [37], and UDDI [113]. SLP provides an example of an attempt by a standards body to define an SDM; Jini provides an example of an inbuilt SDM system in SOA middleware; and UDDI provides an example of an SDM currently in vogue.

3.2.1 SLP

SLP (Service Location Protocol) [49, 50] is an IETF (Internet Engineering Task Force) standard for service discovery. It defines an abstract SDM architecture, of which several implementations exist [7, 87]. The architecture is specified in terms of User Agents (UAs), Service Agents (SAs) and Directory Agents (DAs). Despite this terminology, the three types of "agent" essentially correspond to consumers, services and registries, and the explanation of SLP will be given in these original terms.

A service is advertised in a registry using a registration message. The message contains the service URL (Uniform Resource Locator), a set of attributes, and the lifetime of the advertisement. The URL contains the type and network-address of the service. The set of attributes describes various aspects of the service. Ideally, the set should conform to those attributes defined in a template registered with IANA (Internet Assigned Numbers Authority). Templates [48] are an attempt to standardise service descriptions. The template for a particular service type specifies the attributes used to describe a service of that type, including their default values and interpretation. The advertisement lifetime acts as a lease, defining how long the service advertisement will remain in the registry; a service should periodically refresh its registration if it will be available for longer than a single lease lifetime. Bettstetter and Renner [7] provide an example SLP URL and template advertisement associated with a printer service:

```
service:printer://lj4050.tum.de:1020/queue1
scopes = tum, bmw, administrator
printer-name = lj4050
printer-model = HP LJ4050 N
printer-location = Room 0409
color-supported = false
```

```
pages-per-minute = 9
sides-supported = one-sided, two-sided
```

A consumer submits a service request to the registry in the form of a query which specifies the type and attributes of the required service. The query can be quite flexible, as the required attributes can be specified as an LDAPv3 filter [57] which supports logical operations, wildcards, inequality and substring match. The registry identifies those advertisements which match both the specified type and the attribute filter, and returns the set of corresponding service URLs to the consumer. In terms of the previous example, a consumer query specifying service type "service:printer" and attribute filter "(&(printer-location = "Room 04*")(pages-per-minute > 7))" should match the advertised printer, along with other fast printers on the 4th floor of the building. Finally, the consumer can choose which of the matched services to utilise, and can bind to and interact with the selected service using the corresponding URL.

3.2.2 The Jini Lookup Service

Jini [37] is an SOA middleware solution, developed by Sun Microsystems, which builds on the Java programming language. As such, a Jini service is implemented as a Java object, remotely accessible through a Java interface of methods using RMI (Remote Method Invocation). The inbuilt service discovery mechanism is centred around the Jini version of a registry, known as the Jini Lookup Service.

A service advertises itself in the registry by submitting an RMI proxy stub, which implements the same interface as the service, along with other "attribute" objects. As in SLP, a service advertisement is leased, and will only remain in the registry if the lease is renewed on a periodic basis. A consumer submits a service request to the registry in the form of a template which specifies the Java interface type and attribute objects of the required service. The registry identifies those advertisements which match both the specified interface type and attributes, and returns the set of corresponding service proxy stubs to the consumer. In contrast to SLP, service matching in Jini is a simple process of exact matching, with the consumer-submitted template objects being matched against the service advertisement objects according to Java equality rules. Finally, the consumer can choose which of the matched services to utilise, and can bind to and interact with the selected service using the corresponding RMI proxy stub.

3.2.3 UDDI

UDDI (Universal Description, Discovery and Integration) [113] is an abstract SDM specification defined by the industry-led OASIS consortium. Heralded as the service discovery mechanism for Web Services, there is an expectation [26] that a service will be defined using WSDL, advertised and found using UDDI, and interacted with using SOAP. The UDDI specification defines an XML-based data model for representing services within a registry, and a set of SOAP messages for interacting with the registry. Various companies, such as Microsoft, Sun and IBM, have developed UDDI registry implementations.

Having been developed by industry, the UDDI data model uses business-oriented terminology. Despite this, UDDI places no constraints on its usage, and can be used to advertise services provided by any party, business or otherwise. A service provider is represented by a "businessEntity", which contains information such as the provider name, web-page and contact information. Within a businessEntity are references to "businessServices", which represent the different types of service offered by the provider. Within a businessService are references to "bindingTemplates", which represent the actual service instances that provide the businessService. A bindingTemplate contains the technical information required to utilise a particular service. It contains the network-address of the service, and references to technical specifications with which the service complies. Technical specifications, such as a communications protocol or service interface, are first registered with the UDDI registry as "tModels" (technical models). If a service supports a particular technical specification, its bindingTemplate should refer to the corresponding tModel.

A service provider first registers itself in a registry as a businessEntity. Provider services can then be advertised in the registry by registering the service type as a businessService, and the actual services as bindingTemplates. Each bindingTemplate should reference the tModel of the interface that the service supports. All entities in the UDDI data model, businesses and services, can be tagged with arbitrary attributes in the form of name-value pairs. Indeed, UDDI explicitly supports three standard taxonomies which can provide entity classification attributes: the North American Industry Classification System (NAICS) for business classification, the Universal Standard Products and Services Code System (UNSPSC) for service classification, and the International Organization for Standardization Geographic taxonomy (ISO 3166) for geographic classification. Figure 3.2 shows an example of Web services advertised in a UDDI registry. The ServiceProviderInc company has advertised two services, one in the USA, the other in Europe. Both services

support the ExampleService WSDL interface, and are accessible through the specified URL access-points.

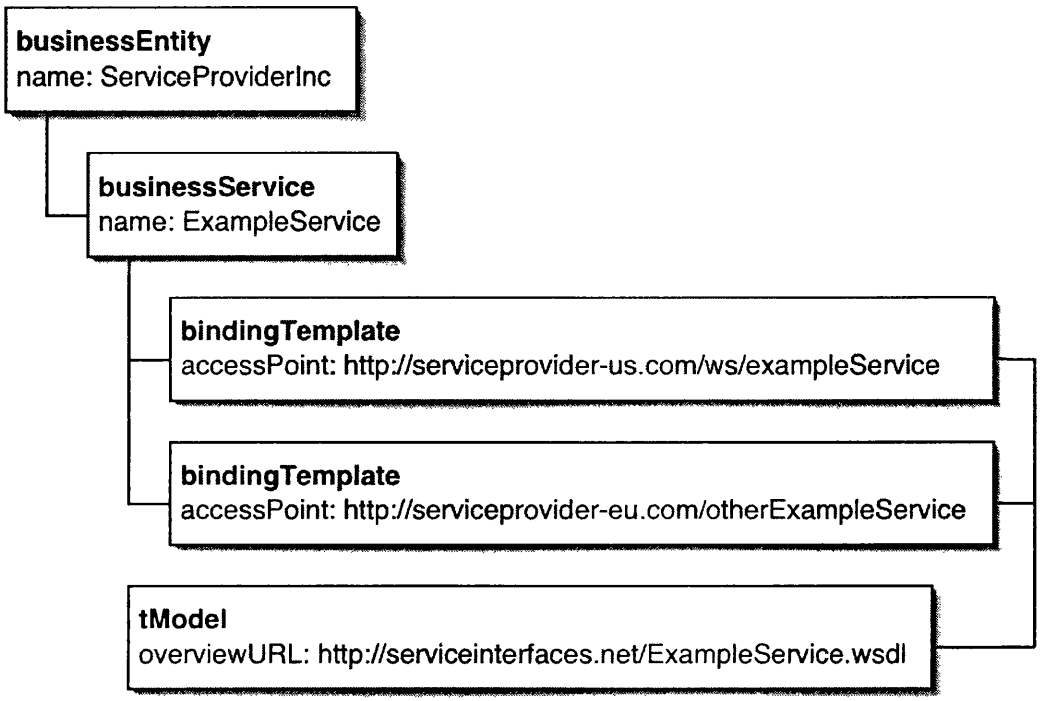


Figure 3.2: An Example of Advertised UDDI Services

In order to request services of a particular type, a consumer submits a query to the registry which specifies the tModel of the required service interface, together with other required attributes [70]. As in Jini, the UDDI registry uses exact matching to identify those advertisements which match both the specified tModel and other attributes, and returns the set of corresponding service bindingTemplates to the consumer. In terms of the previous example (Figure 3.2), a consumer query specifying the ExampleService WSDL tModel should match the two advertised services. Finally, the consumer can choose which of the matched services to utilise, and can bind to and interact with the selected service using the corresponding bindingTemplate.

3.2.4 An Overview of Service Discovery Mechanisms

As demonstrated by the example systems described above, most SDM systems share commonality not just with regard to general structure and operation, but also in the more

specific areas of service description and matching. A service is generally modelled in a registry as a structured, machine-manipulable description, which specifies the service type and a set of attributes detailing different aspects of the service. An attribute is commonly structured as a name-value pair, with the name identifying the aspect under consideration, and the value specifying the state of the service in terms of that aspect. If the attribute is specified using a particular classification scheme, the name will refer to the scheme, and the value to an element within that scheme. Identification of matching services is generally done through a process of exact matching, with each service's advertised type and attributes being tested for equality against the requested service type and attributes (although SLP does provide marginal syntax-matching flexibility).

3.3 An Analysis of Existing Service Discovery Mechanisms

In terms of addressing the key issue of how an appropriate service is dynamically selected for use in an application, existing SDM systems do provide certain benefits to the consumer. When requested by a consumer for services of a particular type with particular attributes, an SDM registry does return an unordered set of services which match those characteristics. This service set can be of value to the consumer in choosing which service to use. It provides him with an awareness of available services that might otherwise be unknown to him, and gives the network-address information required to interact with any of them.

However, there are inherent problems associated with the consumer-driven nature of existing SDM systems, whereby the onus is on the consumer both to define precisely what he requires and then to choose from the returned service set. For example, a consumer must be relatively well-informed in order to devise a request that will identify a set of potentially appropriate services. Firstly, the consumer must be aware of how the type of service that he requires is modelled in the registry. Secondly, the consumer must have an understanding of the important criteria by which service appropriateness should be judged, and of the specific service type model attributes which correspond to these criteria. Thirdly, the consumer must know which specific values of the "criteria" attributes are appropriate, given his particular circumstances. Only then can the consumer devise a registry service request that identifies a set of potentially appropriate services: services of a particular type with particular attributes that have particular values.

For example, the informed consumer who generated the SLP request for a printer in

Section 3.2.1 must have understood how a printer type was modelled in SLP, and have considered that the important criteria by which printer appropriateness should be judged were location (attribute name of "printer-location") and speed (attribute name of "pages-per-minute"). Moreover, he must also have realised that printers on the 4th floor of the building (attribute value of "Room 04*") with a speed greater than 7 pages per minute (attribute value > "7") would be appropriate for his particular circumstances, given his current location and presumable need to print a large document urgently.

Unfortunately, a consumer will not always have this level of knowledge and understanding. He is always likely to know the type of the service required, and perhaps how the type is modelled in a registry. However, when in unfamiliar circumstances with unknown services available, a consumer may not have an informed notion of what service appropriateness is, or how to characterise it in a request: i.e. which service type model attributes with particular values would define an appropriate service? Clearly, an uninformed consumer will have difficulty in devising a request that identifies potentially appropriate services. Even if a request is devised based on some understood criteria and values, the returned set of matches is likely to contain some inappropriate services. Moreover, if the uninformed consumer does not specify certain attribute values, some appropriate services may not be identified.

Consider the previous scenario outlined in Section 2.3.2 in which Alice is attempting to print in the university department, which she has never previously visited. Alice may logically consider that location is an important criterion in judging printer appropriateness. However, although she may know how to specify the printer type location attribute in a service request, she may not know *which* location values to specify: which printer locations would be appropriate, given her current circumstances? Moreover, Alice may be unaware that her visitor role constrains her to using certain "public" printers. This would mean that any printer request that Alice did devise could return inappropriate private printers; the role-accessibility of a printer may have been modelled as a printer type attribute, but this would be of little use to Alice if she is unaware of its importance.

Similarly, in Section 2.3.3, when Bob first assembles his animal genome analyser, he may consider that an appropriate DNA sequence-analyser service is simply a cheap one, and correspondingly devise a request that matches analyser services with an inexpensive price attribute. If he were more informed, he might realise that another significant criterion for judging analyser appropriateness was processing-throughput, given the magnitude of

experimental data involved. In contrast to Bob's actual request, a request matching on price and throughput attributes would have identified a more appropriate set of analysers.

Even if a consumer does devise a registry service request to the best of his ability, it may still be a challenge for him to select the most appropriate service from the returned unordered set of matches. The returned services could be considered equally appropriate, but only in that all match the specified service type and attributes. In terms of other criteria, some services are likely to be more appropriate than others. For example, some might be more reliable or provide better quality. The consumer may have been informed enough to define a request for appropriate services based on certain criteria, but selecting the most appropriate match could still prove difficult and time-consuming if he has no knowledge of the services returned. In order to make an informed judgement, a consumer might expend time and effort investigating all of the different matches, but this could prove frustrating or even impossible if the set was significantly large. Alternatively, a consumer might simply choose a matched service randomly, and risk the consequences.

The effectiveness of a service discovery mechanism based on consumer-specified type and attribute matching therefore depends very much on the knowledge and understanding of the consumer. In the case of an informed consumer, this style of consumer-driven service discovery could provide a flexible tool for the identification of potentially appropriate services that have certain defined attributes. On return of the service set, the informed consumer could theoretically assess the different choices, and select the most appropriate service to utilise.

In contrast, the uninformed consumer might view this style of SDM from a somewhat different perspective. Having limited understanding of what attributes an appropriate service would have, the consumer might struggle to devise a service request. Even if a request was devised based on some understood criteria and values, the consumer would then face the challenging task of identifying what appeared to be an appropriate service from the unordered set of matches.

In view of this, current service discovery research cannot be said to have adequately solved the problem of how an appropriate service can be found and selected. The benefits outlined at the beginning of this section do not necessarily extend to uninformed consumers, who are handicapped by their own lack of knowledge and understanding when using a consumer-driven SDM. Clearly, service discovery needs to change: it should be improved and augmented in order to play a more active role in helping such consumers.

3.4 Summary

The issue of service selection has been discussed in this chapter, and current research in the area, with its focus on service discovery mechanisms, has been considered. The fact has been highlighted that most developed SDM systems share a basic functional commonality, in the style of an automated "Yellow Pages" directory, despite the apparent technical differences between them. This shared commonality has been further demonstrated through a discussion of three frequently-cited SDM systems: SLP, the Jini Lookup Service and UDDI.

The current consumer-driven approach to service discovery has been considered, with its benefits acknowledged and its inherent problems identified. The fact has been noted that the effectiveness of a consumer-driven service discovery mechanism depends very much on the knowledge and understanding of the consumer using it.

It has been argued that service discovery needs to change, better to support the uninformed consumer. A potential way of providing more support will be explored in the following chapter.

Chapter 4

Towards Personalised Service Recommendation

In this chapter, the concept of "personalised service recommendation" is introduced and a proposal given for a style of SDM recommending registry that generates such recommendations. Research relevant to the concept is then detailed and an analysis given of this research. Finally, my research aim in connection with registry-generated personalised service recommendation is motivated and defined.

4.1 From Service Discovery to Service Recommendation

4.1.1 A Proposed SDM Recommending Registry

If uninformed consumers cannot adequately make use of current consumer-driven SDM systems to find and select appropriate services, what can be done to help them? How could service discovery change in order to play a more active role in supporting their service selection? Surely, rather than merely enabling a consumer to discover services that match his defined requirements, an SDM registry should recommend services that seem appropriate. In essence, a registry should aim to provide *personalised service recommendations.*

It is proposed that, in order to provide such personalised service recommendations, an SDM recommending registry would operate as follows. On receipt of a service request, which would state the required service type and any specific attributes, the registry would

first identify *all* available type-matching services, through assessment of the advertised service descriptions. Consumer-specified attributes would be ignored at this stage to allow the registry itself complete control over the identification of all potentially appropriate services. The registry would next determine the appropriateness for the requesting consumer of each of these services, according to some internal logic. Finally, the registry would order the available type-matching services by their perceived appropriateness, ranking them from most to least appropriate. This resultant ordered list would be the personalised service recommendation and would be returned to the consumer. Those services which were highly ranked would be those considered the most appropriate for the consumer's particular needs and circumstances. Such a style of recommendation is conceptually similar to the relevance-ordered list of documents returned by a Web search engine. If the highly ranked services were truly appropriate choices, enabling the uninformed consumer to search through the recommendation and select an ideal service with minimum time and effort, then the recommendation would have been truly effective. The recommendation would also be filterable, to show only those services that matched the consumer-specified attributes.

Clearly, in order to determine personalised service appropriateness, a registry would need to take into consideration relevant aspects of the consumer's particular circumstances: it would need to be *context-aware* [86]. Such considered contextual information might include, for example, the state of the consumer, that of the surrounding computing and physical world, and the state of the considered services. This information might be statically defined in, or dynamically sensed by, the registry itself, or obtained from outside sources such as the requesting consumer or the services themselves.

Such a proposed form of SDM recommending registry should alleviate the problems currently faced by the uninformed consumer when using a consumer-driven SDM system. This is because, with a proposed recommending registry, the burden of identifying appropriate services would move from the consumer himself to the registry, which would perform the task autonomously. If a registry-generated personalised service recommendation was effective, the consumer would be able to search through it and select a truly appropriate service with minimum time and effort.

Figure 4.1 provides a pictorial example of a proposed SDM recommending registry in operation. A consumer has submitted a service request to the recommending registry, stating the required service type and any specific attributes. The registry has identified

those advertised services with the required type, namely A, B, C, D, E and J. Having assessed contextual information acquired from various sources, the registry has then determined the appropriateness of each of these services. Finally, it has ranked the services by appropriateness, and returned the resultant list to the consumer as a personalised service recommendation: D has been ranked first (most appropriate), J and C joint second, A third, and B and E joint last (least appropriate). If the consumer applied the filter, the recommendation would show only J and A (shown in italics), which both match the required attributes, ranked first and second respectively. Obviously, in a real-world setting, the number of services involved could be much greater.

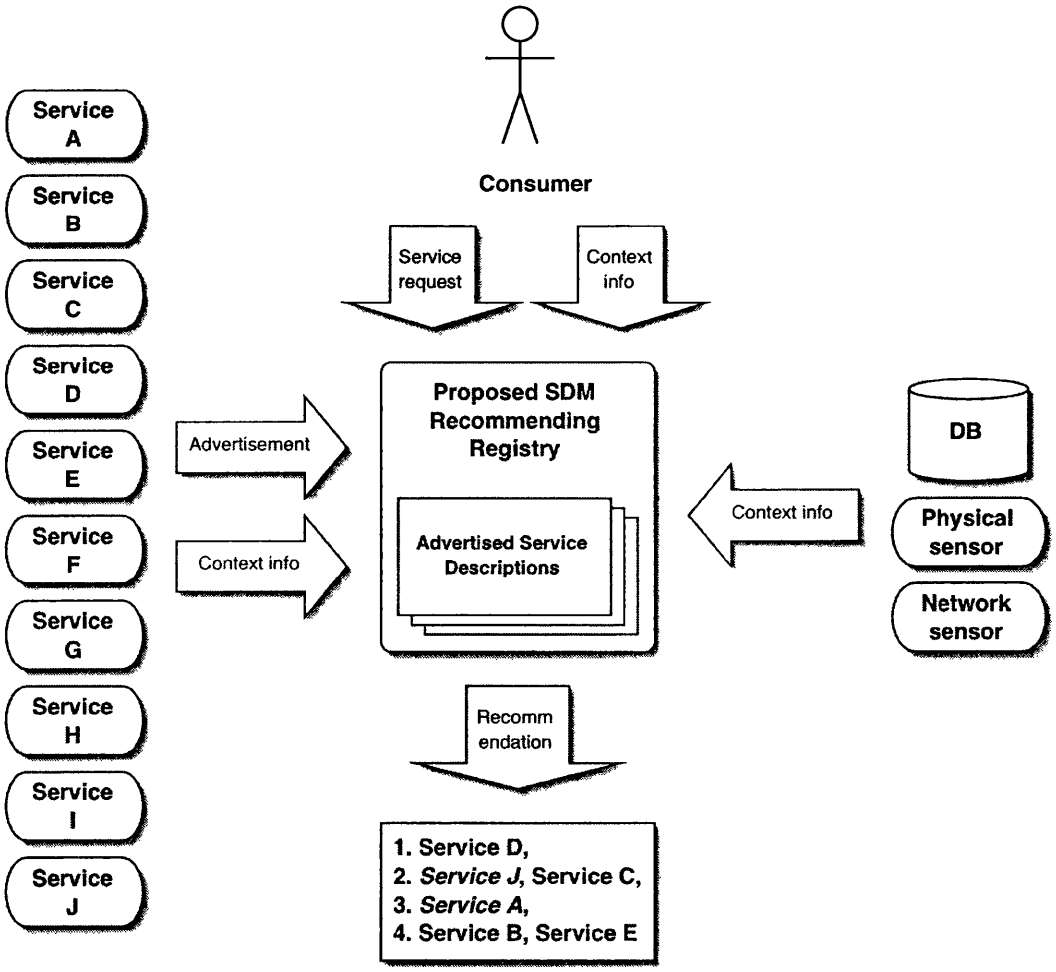


Figure 4.1: A Proposed SDM Recommending Registry in Operation

The running examples of Alice and Bob begun in Section 2.3 can be used to illustrate how such a proposed form of SDM recommending registry might benefit uninformed

consumers. Alice, for example, could submit a printer request to the SDM recommending registry of the visited university department. This registry would then identify all available departmental printers and determine their appropriateness, taking into consideration relevant contextual information. Such information could include Alice's location and role, and each printer's location, role-accessibility, capability, reliability and load. The registry would then ideally generate an effective personalised service recommendation for Alice, which ranked highly those departmental printers that were nearby, publicly-accessible, lightly-loaded, and able to print documents of good quality both reliably and quickly. In response to a DNA sequence-analyser request made by Bob, a grid SDM recommending registry might take into consideration Bob's preference for an inexpensive service, the network conditions, and each available analyser's cost, reliability, accuracy, load and processing-throughput level when determining analyser appropriateness. The registry would then ideally generate an effective personalised recommendation for Bob, which ranked highly those DNA sequence-analysers that were cheap, lightly-loaded, had low latency and high bandwidth, and were able to process genome data quickly, accurately and reliably.

The examples of Alice and Bob also serve to highlight the fact that the problem of service selection is a universal one. Alice requires a printer in the departmental ubicomp environment, whilst Bob requires a DNA sequence-analyser within a grid. Different service types, different scenarios, but the same problem of finding and selecting an appropriate service from the many available within a service-oriented architecture. Given the expected growth of such service-oriented architecture in areas as diverse as the business world, the services web, ubiquitous computing, and grid computing, as outlined in Chapter 2, the need for a generally applicable solution to the general problem of service selection becomes apparent. Thus, any research into the generation of personalised service recommendations through the use of such a proposed form of SDM recommending registry must be undertaken from this perspective of generality.

4.2 Existing Research of Relevance to the Proposal for an SDM Recommending Registry

Although, theoretically, the notion of registry-generated personalised service recommendation would seem to evolve naturally from the current service discovery approach, there

appears to be surprisingly little specific in-depth research being undertaken. Indeed, the concept does not appear to be the subject of any dedicated research area, and the term "personalised service recommendation" has needed to be coined in this thesis in order to define the idea more precisely.

Despite this, some research has been undertaken which would seem to fall within my remit of personalised service recommendation research, though being presented from other perspectives. For example, there is some evidence of an awareness that consumers would benefit from more personalised support in service selection, and some researchers have noted the value of using contextual information in the area of service discovery [5,53,67,69,72,80,107,123]. Some practical research has also been undertaken which does relate in some ways to the proposal for an SDM recommending registry. However, no concrete systems that actively provide recommendations appear to have been deployed in a real-world setting.

An overview of this related research is given below. For the sake of clarity, what are, in fact, disparate pieces of research have been grouped into two distinct categories. The first category consists of generic technical mechanisms that could enable the construction of a style of SDM recommending registry. The second category comprises various SDM-style systems which generate a form of personalised service recommendation based on very specific notions of service appropriateness. Despite this imposed categorisation, it should be pointed out that, in reality, the research projects described are independent of one another, no connections appear to exist between them, and associated project research papers do not reference one another.

4.2.1 Generic Technical Mechanisms

Extensions to SLP

The concept of returning matched services to the consumer as an appropriateness-ordered list has been posited as an improvement to an existing consumer-driven service discovery mechanism. RFC 3421 [122] proposed an extension to SLP in which matched services would be returned as a ranked list, sorted on a particular service attribute defined by the consumer. Hughes et al [59] have also proposed a similar style of extension to SLP.

However, these proposals would still be of little value to the uninformed consumer, as they demand an understanding of which single provider-defined service attribute is of most importance in judging service appropriateness, on which the sort will then be based. Fur-

thermore, the required direction by the consumer precludes any informed recommendation being autonomously generated by the registry.

INS

An SDM system that could autonomously provide registry-generated service recommendations was INS (Intentional Naming System) [1], developed in 1999. Although primarily focused on addressing technical issues such as scalability and fault tolerance, INS did introduce the concept of a generic, provider-specified service "metric". This numeric value was intended to represent the current appropriateness of a service for consumers, and was supplied by the service provider as part of a registry service description; if the appropriateness of a service changed, the provider was correspondingly expected to update the description. The metric information of matched services was returned to the consumer, and it was suggested that the consumer should choose the service with the smallest metric (where a smaller metric meant a better service).

However, although the metric concept did introduce the notion of service appropriateness into a service discovery mechanism, a recommendation based on it would not be personalised towards a particular requesting consumer. Rather, there was an implied assumption that a service with a specific metric would have the same level of appropriateness for all consumers, despite their differing circumstances. Moreover, the onus of objective evaluation and updating of service appropriateness rested with the service providers themselves, and was thus based on the debatable expectation that the providers would always prove reasonable and reliable.

Context Attributes

More recently, Lee and Helal [72] have modified the Jini Lookup Service to allow advertised service descriptions to contain dynamic "context attributes". Similar to an INS service metric, a context attribute is a numeric service appropriateness value, again specified by the service provider. However, the context attribute is implemented as a Java object, which returns the appropriateness value on execution. Once a Jini registry has identified those services matching a consumer's request, the context attribute of each service is executed. The matched services are then ordered by the returned values (most to least appropriate), and the corresponding list is returned to the consumer as a service recommendation. Since a context attribute is dynamically evaluated at request-time, the returned value

might better reflect the current appropriateness of a service than a statically-defined INS service metric. Lee and Helal provide an example [72] in which an executed context attribute remotely checks the current load of its represented service, and constructs a service appropriateness value based on the result.

However, it is not clear how personalised a service recommendation based on this approach could be, since no contextual information explicitly referring to the requesting consumer is used in generating a recommendation. It is suggested that if a consumer contacts a registry which is nearby in physical and network terms, then the state of this registry (where the context attributes are evaluated) will serve as an approximation of that of the consumer. However, since a registry can only really approximate the state of a consumer in terms of his physical and network location, personalised service appropriateness can only be calculated in terms of these specific contextual aspects. For example, this approach would be unable to recommend only publicly-accessible printers to Alice, as a registry would have no awareness of Alice's role.

Semantic Web Services

The active research area of "Semantic Web Services" [79, 80, 109] is also partly concerned with developing an approach which could enable an SDM registry to identify appropriate services autonomously for the consumer, but in a completely different manner. Semantic Web Services (SWS) are motivated by the Semantic Web vision promoted by Tim Berners-Lee, in which the World Wide Web evolves towards a state where "information is given well-defined meaning, better enabling computers and people to work in cooperation" [6]. In essence, the Semantic Web is concerned with describing the semantics (meaning) of web-accessible resources, such as web-pages and web services, using unambiguous, machine-understandable, explicitly-defined meta-data.

Researchers claim that the Semantic Web should enable the construction of more useful, intelligent and autonomous programs, as software will be able to "understand", reason and infer over manipulated resources using the corresponding meta-data. In terms of services, researchers claim that "the Semantic Web should enable users to locate, select, employ, compose and monitor Web-based services automatically" [18]. Consequently, one aspect of Semantic Web Services research is the development of SDM registries that could autonomously generate service recommendations through the assessment of detailed service meta-data; with such a recommendation, a user's representative agent might perhaps select

and utilise an appropriate service.

SWS research has focused primarily on the development of markup languages. These languages should, theoretically, be able to describe services in a richer, more precisely-defined and standardised manner than is currently possible using a UDDI service description. These languages, such as OWL-S (Web Ontology Language - Services) [18] and DAML-S (DARPA Agent Markup Language - Services) [17] are based on the concept of an ontology. An ontology provides a common vocabulary for a particular body of knowledge, specifying the meaning of and logical relationships between concepts, together with associated inference rules. OWL-S and DAML-S provide generic ontologies for services, which contain a set of concepts and relationships for describing the type and attributes of a service.

Certain researchers [2,88,92] have developed augmented UDDI registries to enable services to be advertised in terms of these languages. A consumer phrases his service request in terms of an ontology markup language, which is matched against the correspondingly advertised services. However, in contrast to normal UDDI exact matching, there is an expectation that an ontology-based registry will somehow be able to match services on a semantic level, in terms of ontological concepts, relationships and inference rules, enabling a greater understanding of services' appropriateness. Pokraev et al [93] have also modified a UDDI registry to represent required contextual information in terms of DAML-OIL, another ontology markup language, with ontologies being used to represent different contextual domains such as a consumer's location, time, social and physical conditions.

Despite this initial activity, however, the main focus of SWS research continues to be on the development of languages and methods for the ontological representation of services. Little research has currently been done into how exactly service appropriateness would be determined through assessment of such service descriptions.

4.2.2 SDM-style Recommending Systems

In contrast to INS, Context Attributes and Semantic Web Services, which all focus in their different ways on developing basic generic technical mechanisms, there are a small number of SDM-style systems which focus instead on generating personalised service recommendations through the consideration of contextual information. However, with such systems, services are recommended according to only very specific notions of appropriateness: physical location only or network performance only. For systems that consider

physical location, the nearer a service is to a consumer, the more appropriate it is. For those considering network performance, the better the perceived network performance of a service, the more appropriate it is for a consumer. Two such systems of each type are discussed below.

Location-based Service Appropriateness

Websigns Websigns [96] was part of the Cooltown project [65], which developed ubicomp system infrastructure that could associate Web computing resources with real-world physical entities. Cooltown researchers envisaged people equipped with mobile computing devices accessing resources associated with entities in their surroundings, such as nearby restaurants, theatres and historical sites. The Websigns system aimed to recommend appropriate resources (known as "e-services") to a user, based on his location. Each service was detailed in a WsML ("Websign Markup Language") description containing its type and URL network-address, the location that it was associated with (as a latitude / longitude pair), and a range over which it was applicable; a service was then advertised under this description in a WsML server, which essentially acted as a form of SDM registry.

The Websigns client running on a user's mobile computing device occasionally queried a WsML server, specifying the user's location sensed using GPS (Global Positioning System). The server assessed the advertised descriptions to identify those services that were relatively close to the user's location, in terms of Euclidean distance, and the descriptions of these nearby services were returned. The client further filtered these services on their descriptions, identifying those that the user was currently in range of (again using the sensed location), and was currently facing (using compass-sensed orientation). This filtered list of appropriate services was then presented to the user as a form of personalised service recommendation. The Websign developers decided to perform the final service-filtering operations on the client, but these could have also been performed in a WsML server registry if it had been supplied with the relevant contextual user information.

MobiShare The more recent MobiShare middleware architecture, developed by Valavanis et al [114], is also driven by a similar vision to that of Cooltown. MobiShare has been designed to support an ubicomp environment in which Web Services are associated with aspects of the real world. As with Cooltown, users are envisaged accessing services through mobile computing devices, but emphasis is also placed on the mobility of the ac-

tual services themselves. Some services are expected to be hosted on the mobile computing devices of certain users, who will offer them to others. For example, one such provider might be a freelance taxi-driver, offering his trade as a Web Service.

To support this scenario, MobiShare structures the world as a collection of cells, each mapping onto a wireless access point. A cell is controlled by a Cell Administration Server (CAS), which acts as a form of SDM registry. When a service becomes available in a cell, it is advertised in the controlling CAS under a description containing the service WSDL type, and a classification of the service in terms of an ontology of service categories (e.g. city guide, theatre ticket reservation, taxi booking, etc). When a user requires a service, he submits a request to the local CAS, specifying the type or category of service required; the request also contains the user's sensed location and orientation. The CAS first identifies those type-matching services in the current cell, which must be near to the requesting user (being in the same cell). If the user's location and orientation indicate that he is moving towards the edge of a cell, the CAS also forwards the service request to the CAS that manages the adjacent area. Finally, the list of nearby type-matching services, registered in the current and adjacent cells, is returned to the user as a form of personalised service recommendation.

Network-performance based Service Appropriateness

NSSD Huang and Steenkiste refer to the problem of generating service recommendations based on network performance as "Network-Sensitive Service Discovery" (NSSD) [58]. They have developed an implementation of SLP which augments a registry with network measurement capability. A consumer submits a service request to the registry as a normal SLP query, but can also request that services be assessed in terms of latency, bandwidth or server load. The registry identifies matching services as normal, but then uses its network measurement capability to determine those services with good performance from the perspective of the consumer's network location: ideally those with low latency, or high bandwidth, or low server load. Well-performing matching services are then returned to the consumer as a form of personalised service recommendation.

Xu et al Another SDM system that took network performance into consideration was proposed and simulated by Xu et al [120]. They proposed that when a consumer selected a service using an SDM, and then utilised it, the experienced network Quality of Service

should be monitored. This QoS level information would then be fed back into the SDM, where it would be stored in the service's advertised description. As with the appropriateness values of INS and Context Attributes, no information is given as to what exactly a QoS level would represent, only that "the higher the QoS level, the better the QoS observed". A registry would then use these recorded past QoS levels of services to determine their current appropriateness, and could then generate a service recommendation. However, it is not clear from the description of the research exactly how service appropriateness would be determined through the assessment of associated QoS levels.

4.3 A Discussion of Existing Relevant Research

Although all of the research detailed above focuses on the development of systems that generate forms of service recommendation, the individual projects themselves had very specific and very different initial motivations. For example, Context Attributes specifically aimed to improve the SDM element of Jini, the SWS research is motivated by a strong belief in the value of ontological service descriptions, and Websigns was developed specifically to help people find physically nearby services in an ubicomp environment. No-one seems yet to have comprehensively explored the concept of registry-generated personalised service recommendation from the perspective of its being a generally-applicable solution to the general problem of service selection.

It could be argued that INS, Context Attributes and SWS research, the three research projects categorised together as generic technical mechanisms, were focused on generality in their own different ways. Theoretically, such mechanisms do provide a generally applicable basis on which to construct any recommending registry, regardless of the particular deployment scenario or service types involved. Indeed, a generic mechanism should make registry development quicker and easier. However, apart from imposing a very basic structure and a form of knowledge representation on a recommending registry, it is questionable how much value the three considered mechanisms could have in terms of simplifying registry development. This is because none provides any general support for addressing arguably the most challenging issue in service recommendation generation: how is service appropriateness determined?

In the case of INS and Context Attributes, the determination of service appropriateness is placed in the hands of the service provider through the use of a service appropriateness

metric. Although such an approach significantly simplifies the functionality required of a recommending registry, certain problematic issues arise. Firstly, how exactly is service appropriateness determined, and is a service provider the best party to decide what is appropriate or not for a consumer? Secondly, since no explicitly specified information about the requesting consumer is taken into consideration during determination of service appropriateness, how personalised can a recommendation really be?

In the case of Semantic Web Services, research into service recommendation is very much at an early stage. Ontology markup languages such as OWL-S may yet provide a standardised and powerful way of defining and manipulating service descriptions and other contextual information within a recommending registry. However, there is a marked difference between merely defining relevant knowledge and actually using it.

Those research projects categorised as SDM-style recommending systems, namely Websigns, MobiShare, NSSD and Xu et al, ignore generality altogether. Each project is specific to a particular type of scenario. Websigns and MobiShare were both designed for ubicomp environments, while NSSD and Xu et al were designed for use in more traditional distributed computing networks. Thus, whilst all the systems do address the issue of determining service appropriateness through consideration of contextual information, they do so only in terms of the particular criteria which the researchers considered important in the anticipated deployment scenarios. In the case of Websigns and MobiShare, the most important criterion for service appropriateness in an ubicomp environment was considered to be physical location. More specifically, the nearer a service, in terms of Euclidean distance, the more appropriate it is perceived to be. For NSSD and Xu et al, to be used in traditional distributed computing networks, the criterion of prime importance was seen as network performance. Although there is a lack of detail with Xu et al, NSSD uses either the criterion of network latency, or bandwidth, or service load to determine service appropriateness, but only singly and not in combination.

It should also be noted that although, when deployed in their intended scenarios, these SDM-style recommending systems might well generate adequate recommendations, they are inherently constrained in their effectiveness by the very fact that they were designed to determine service appropriateness in terms of a single particular criterion alone. Perhaps the ideal benchmark against which a service recommender should be compared is an informed human being. When deciding which service to choose, an informed person would consider service appropriateness in terms of a range of relevant criteria. Depending on sce-

nario and service type required, he might well consider location or network performance, but might also consider, for example, cost, capability, reliability, quality, reputation, accuracy and much more. Only when service appropriateness is able to be determined in terms of a range of relevant criteria can truly effective personalised service recommendations be generated.

4.4 Aims of Research

As the discussion of relevant research has made clear, there is definite interest, albeit in an uncoordinated form, in the idea of personalised service recommendation. However, no comprehensive research appears to have been undertaken which addresses the issue of registry-generated personalised service recommendation from the perspective of ensuring that the resultant research findings are generally applicable. More precisely, no-one has yet addressed the question which would seem to be of the utmost importance, namely:

> What general design for an SDM recommending registry would enable the generation of effective personalised service recommendations?

A well-devised general design would be of considerable value because it could provide a blueprint for the construction and operation of any SDM recommending registry, regardless of service-oriented deployment scenario and service types involved. Moreover, a successful, detailed design should allow an effective corresponding generic mechanism to be developed. Such a generic mechanism would significantly simplify the construction of any recommending registry by providing a solid technical infrastructure on which to build one. A detailed design should address the challenging issue of service appropriateness determination, and should thus enable the corresponding generic mechanism to provide more effective general support than is currently provided by INS and Context Attributes.

The aim of my research, therefore, is to attempt to address the question posed above, by exploring potential design options and developing solutions to relevant associated research issues. As the general design is developed, it will be repeatedly tested for effectiveness.

The proposal for an SDM recommending registry defined in Section 4.1.1 will serve as the abstract model which defines general registry operation. To reiterate:

> In requesting a personalised service recommendation, a consumer would submit a service request as input, stating the required type and any specific attributes.

> The registry would first identify all those advertised services of the required type, and then determine their perceived appropriateness through consideration of relevant contextual information. Finally, it would rank these available type-matching services by their appropriateness, and return the resultant ordered list to the consumer as a personalised service recommendation. The recommendation would also be filterable to show only those services that matched the consumer-specified attributes.

Development of the general design will be based on this abstract model.

In order to initiate the investigative process, techniques from the described SDM-style recommending systems will be assessed for their applicability to a general recommending registry design. Despite criticism of these systems, it must be acknowledged that they have succeeded in generating a form of personalised service recommendation through consideration of some contextual information. The associated techniques could consequently be of use. In contrast, the described generic technical mechanism research is of little use in terms of this investigation, as it provides no guide to service appropriateness determination.

4.5 Summary

In this chapter, the concept of providing better support for the uninformed consumer through personalised service recommendation has been considered, and a style of SDM recommending registry to generate such recommendations proposed. It has been argued that, given the universality of the service selection problem, any research into registry-generated personalised service recommendation should be undertaken from the perspective of generality.

It has been noted that there seems currently not to be any dedicated research area concerned with the concept. However, research that is of relevance has been identified, categorised into the two groups of generic technical mechanisms and SDM-style recommending systems, detailed, and discussed.

The observation has been made that, despite interest in the idea of personalised service recommendation, no-one has yet attempted to address the specific question of what general design for a recommending registry would enable the generation of effective recommendations. My research aim of attempting to address this question has been stated, and an abstract model of an SDM recommending registry, which will serve as the basis

for general design investigation, defined.

Finally, it has been noted that techniques used in the SDM-style recommending systems outlined could be of use in furthering the research aim. These techniques will thus be used in the next chapter to derive a theoretical general recommending registry design.

Chapter 5

A Theoretical General Design for a Recommending Registry

In this chapter, a theoretical general design for a recommending registry, derived from techniques used in the basic SDM-style recommending systems discussed in Chapter 4, is presented. The suitability of the theoretical design is subsequently assessed, and the conclusion made that its potential drawbacks outweigh any potential benefits.

5.1 A Theoretical Design

My investigation of a general design for an effective recommending registry commenced with the development of a theoretical design, derived primarily from techniques used in the basic SDM-style recommending systems discussed in the previous chapter. These systems are Websigns [96], MobiShare [114] and NSSD [58]. The research of Xu et al [120] was not considered as it lacked detail. The abstract model of a recommending registry defined in Section 4.4 served as the basis for the theoretical design.

Once such a design was developed, it was assessed for its suitability in terms of a) whether a registry which implemented the design should be able to generate effective personalised service recommendations; and b) whether a corresponding generic mechanism developed to support the design would simplify registry construction. It should be noted that the theoretical design was developed only to such level of detail as enabled this assessment to be made.

Given that the available type-matching services in a personalised service recommen-

dation will be ordered according to their perceived appropriateness for the requesting consumer, how such service appropriateness is determined is arguably the most important issue in recommending registry operation. As was noted in the last chapter, this might involve the consideration of a range of relevant criteria in combination.

In view of this, how might service appropriateness be determined through the use of existing techniques?

5.1.1 World Models, Appropriateness Rules and Proportional Criteria Weightings

As was stated in the abstract model, a recommending registry would need to be context-aware in order to generate personalised service recommendations. The existing SDM-style recommending systems of Websigns, MobiShare and NSSD do take some relevant contextual information into account when determining service appropriateness, having first acquired it from various sources. The registries acquire details of requesting consumers' physical or network locations, either through direct sensing, or through direct submission by the consumers themselves. Additionally, information about available services is supplied to the registries, as usual, in the form of the attribute-based descriptions used by providers to advertise their services. Internally, all this contextual information forms an attribute-based representation, which can be defined as a "world model", describing the state of the requesting consumer and available services. Through assessment of the world model, the registries then determine the appropriateness of a service for the requesting consumer.

In terms of the theoretical design, a registry would similarly need to contain a world model. However, to enable the registry to have comprehensive context-awareness, in order to generate more personalised and more effective service recommendations, a world model richer in content than those currently used in the existing SDM-style recommending systems would be required. For example, the state of the requesting consumer might be described in terms more comprehensive than merely location, and details of the surrounding computing and physical world might also be captured. The additional contextual information might be statically defined in, or dynamically sensed by, the registry itself, or obtained from outside sources.

With this richer world model in place, how would service appropriateness be determined through its assessment? A technique of what can be termed "appropriateness rules" has been used in the SDM-style recommending systems. A rule precisely states how the

appropriateness of a service, in terms of a particular criterion, is determined through the assessment of particular elements in the world model. The three considered systems define a small number of rules but, as was noted in the last chapter, these determine service appropriateness only in terms of either physical location or network performance. For example, the appropriateness rule used in Websigns can be defined as "the smaller the calculated Euclidean distance between the service's physical location (attribute-name = service-location) and that of the consumer (attribute-name = requesting-consumer-location), the greater its appropriateness".

Most of the rules defined in the three systems are conditional, in that the appropriateness of a service calculated using a rule is dependent on aspects of the requesting consumer's particular situation: for example, in the case of Websigns, service appropriateness is dependent on the consumer's physical location. However, the NSSD system also defines an unconditional rule, in which calculated service appropriateness is not dependent on the requesting consumer's situation. This rule determines service appropriateness in terms of service load, and can essentially be defined as "the smaller the service's load (attribute-name = service-load), the greater its appropriateness". As previously stated, service appropriateness might need to be judged according to a range of criteria beyond physical location and network performance in order to generate effective recommendations. In terms of the theoretical design, therefore, a registry would need to contain a range of conditional and unconditional rules which determined service appropriateness in terms of a variety of relevant criteria, utilising the contextual information available in the richer world model.

The process of service appropriateness determination should ideally allow for the consideration of a range of criteria *in combination*, yet none of the considered systems provide this facility. Thus, the final question that needs to be addressed in developing the theoretical design is how such multi-criteria service appropriateness could be achieved. Some recent related research, although emanating more from of an AI perspective, does use a technique that could be applicable. Lee et al [73, 74] have recently proposed and partially implemented a "Personal Router", which aims to aid the user in selecting the most appropriate choice from a market of services in an ubicomp environment. Similar to my concept of a recommending registry, a Personal Router (PR) would attempt to recommend appropriate services to a user. PR research has so far focused on using machine-learning techniques to determine how a user balances, or trades off, service network performance

(quality) against financial cost when judging service appropriateness. When required to generate a recommendation, the PR determines service appropriateness by taking the numeric quality and cost attributes of a service, and combining them into a single appropriateness value by applying the learned trade-off weighting in a weighted sum. For example, if the PR had learned that a user considered cost more important than quality on a ratio of 4:1, then the overall service appropriateness value would be generated as $(0.8 * cost) + (0.2 * quality)$.

Although PR research has considered only network quality and financial cost, a similar weighted sum technique could be used in the theoretical design to generate a single value representing the multi-criteria appropriateness of a service. Service appropriateness could first be determined for different criteria through execution of the corresponding conditional and unconditional rules. The appropriateness values generated by these rules would then need to be converted into a normalised numeric form, e.g. between 0 (completely inappropriate) and 1 (completely appropriate). These different normalised values could then be combined into a single multi-criteria appropriateness value using a proportional weighting. The weighting would represent the contribution (i.e. importance) of each criterion when calculating overall service appropriateness, with the individual weights summing to 1.0. For example, if service appropriateness was judged in terms of physical location, network latency and bandwidth, with location being considered twice as important as the other two, the corresponding weighting would be $[location\_weight = 0.5, latency\_weight = 0.25, bandwidth\_weight = 0.25]$, and overall service appropriateness would be calculated as $((0.5 * location\_appropriateness) + (0.25 * latency\_appropriateness) + (0.25 * bandwidth\_appropriateness))$.

Once the overall appropriateness of each available type-matching service had been calculated, a personalised recommendation could finally be generated by ranking the services according to this value.

5.1.2 Applying the Design

The theoretical design described above is essentially a generalisation and augmentation of the various specialised approaches to service recommendation taken by the existing SDM-style recommending systems. The Websigns, MobiShare, and NSSD recommending registries all use a form of world model, and determine service appropriateness using a small number of rules that assess the model. However, all aspects of their particular approaches

are focused on generating recommendations in terms of a single criterion. The theoretical design relaxes this specialisation, advocating a richer world model, which a wider range of conditional and unconditional rules would assess when determining service appropriateness in terms of a greater variety of relevant criteria. Proportional criteria weighting should make multi-criteria service appropriateness possible. The actual content of a world model, the particular conditional and unconditional rules, and the specific criteria weightings used in a recommending registry would obviously depend on the scenario in which it was deployed, and the types of service that it was required to recommend.

Imagine that the recommending registry in the university department where Alice requires a printer conforms to this theoretical design. The registry would provide recommendations for the different types of services offered by the department. When Alice requests a printer recommendation, the registry would first use the advertised service descriptions in the world model to identify all available printer services. It would then determine printer appropriateness in terms of physical location, role-accessibility and current load, executing the corresponding rules for each matching service. The location and role-accessibility rules would both be conditional, whilst the load rule would be unconditional. The location rule would calculate the corridor distance between Alice's room and that of a printer: the smaller the distance, the greater the appropriateness. The role-accessibility rule would match Alice's role against the role-accessibility of a printer: a matching printer would be defined as being totally appropriate, whilst a non-matching one would be defined as only partially appropriate. The current load rule would assess a printer's current load: the smaller the load, the greater the appropriateness. All of the required attributes - locations, roles, role-accessibility, and loads - would be available in the world model. Having determined the different appropriateness values of each service in terms of the three criteria, the registry would then calculate the overall multi-criteria appropriateness value of services using a pre-defined criteria weighting: location might be given most weight, closely followed by role-accessibility, with the remaining weight given to load. Finally the printer services would be ranked by overall appropriateness, and the resultant ordered list would be returned to Alice.

Or, imagine that the grid recommending registry which Bob subscribes to also conforms to the theoretical design. Obviously the grid registry would have different world model content, rules and criteria weightings to that of the departmental registry. When Bob requests a DNA sequence-analyser recommendation, the registry would identify all

available analyser services, and would then determine service appropriateness in terms of unconditional rules for financial cost, processing-throughput and reliability, and a conditional rule for bandwidth. The unconditional rules would assess the descriptive attributes of a service, with lower financial cost, higher processing-throughput and better reliability being considered more appropriate. The bandwidth rule would estimate potential service bandwidth from Bob's network location, with higher bandwidth being considered more appropriate. Service multi-criteria appropriateness would then be calculated, perhaps with cost and processing-throughput being given most weight, and the remaining weight being split between reliability and bandwidth. Finally, the analyser services would be ranked by overall appropriateness, and the resultant list returned to Bob.

5.2 Assessing the Suitability of the Theoretical Design

Two questions need to be addressed in assessing the suitability of the theoretical design. Firstly, is it probable that a recommending registry which implemented this design would generate *effective* personalised service recommendations? Secondly, to what extent would a corresponding generic mechanism, developed to support this design, simplify the construction of a recommending registry? These two questions are considered below.

5.2.1 Effective Personalised Service Recommendations

Theoretically, a registry that implemented this design should be able to generate effective personalised service recommendations. The use of a world model, appropriateness rules, and proportional criteria weightings could enable service appropriateness to be determined in an informed manner, with an extensive range of criteria being taken into consideration in combination. If the world model was sufficiently accurate and detailed, with truly significant appropriateness criteria specified as well-defined rules, and their actual importance reflected in a proportional weighting, then a recommending registry should be able to generate effective recommendations.

However, it is questionable whether this ideal arrangement could always be attained, owing to three problematic aspects of the design. Firstly, the effectiveness of a registry, and thus of the personalised service recommendations generated, would be dependent on the particular ability and awareness of the registry developer. Secondly, unless occasional manual modifications were made to a registry to reflect relevant changes, the effectiveness

of generated recommendations could deteriorate over time. Thirdly, world model content could well be inaccurate.

Dependency on the Developer

With the theoretical design, the developer of a recommending registry would be wholly responsible for deciding how service appropriateness should be determined, and for encoding the process in terms of the corresponding world model, appropriateness rules and proportional criteria weightings. As such, whether or not a registry could generate effective recommendations would depend significantly on the ability and awareness of the developer, particularly his knowledge and understanding. His subjective opinions would also have an impact.

The development of such a recommending registry could be a complex process. For each type of service to be recommended, the developer would first need to ascertain which relevant criteria should be taken into consideration. He would then need to decide how such criteria should be encoded as appropriateness rules, and what information these rules would require to operate. This information would need to be acquired and then represented in the world model. Finally, in defining the proportional criteria weighting for a particular type of service, the developer would need to determine the relative importance of each of the considered criteria. If the developer made a bad judgement at any stage in this development process, then the effectiveness of recommendations generated by the registry could be severely reduced.

Consider the problems that might occur in the development of the departmental recommending registry used by Alice. In terms of printer services, the registry developer might not be aware that role-accessibility was an important issue when accessing printers. This could result in inappropriate printers in private offices being highly ranked in the recommendation generated for Alice. Alternatively, the developer might be aware that physical location was an important criterion to consider, but might decide that the best way to implement the corresponding rule would be to calculate the Euclidean distance between the sensed height / latitude / longitude locations of the requesting consumer and a printer. This could also result in inappropriate printers being highly ranked, such as those which were physically nearby but difficult and time-consuming to reach owing to the structure of the building. Humans currently lack the ability to walk in any direction through walls and other physical objects! A better version of the physical location rule

might involve the calculation of distance in terms of rooms and corridors, although this would require the world model to contain details of the building's architecture. Finally, if the developer did actually determine printer appropriateness in terms of well-defined rules for physical location, role-accessibility and current load, he might then decide subjectively that load was of most importance when constructing the proportional criteria weighting. In contrast, most consumers might value location and accessibility more, preferring a nearby, accessible printer to a lightly-loaded distant one.

If an external developer was hired to construct a recommending registry for a particular scenario, he might not have an awareness of the issues important in determining service appropriateness. A developer might attempt to address this by making use of questionnaires, interviews or ethnography to ascertain the opinions or behaviour of informed users in selecting an appropriate service. However, even if this information was taken into consideration, whether or not a registry could generate effective recommendations would still depend completely on the ability of the developer to determine service appropriateness, in terms of his encoded world model, rules and weightings.

Given that the ability and awareness of a developer cannot be guaranteed and could vary widely, there is a distinct possibility that a constructed recommending registry which conformed to the theoretical design could generate ineffective personalised service recommendations.

Reflecting Change

Imagine that, for a particular scenario, a developer does construct a recommending registry which generates effective recommendations for different types of service. On initial operation, the world model accurately reflects the current state of the world, and overall service appropriateness is determined in terms of adequately weighted significant criteria specified as well-defined rules. However, over time, changes that affect the appropriateness of services will inevitably occur. The qualities of available services and aspects of the physical and computing environment might change, whilst new appropriateness criteria could become important and existing ones less so. Unless the registry adapts to reflect such changes, through modification of its world model, rules and weightings, the recommendations that it generates will gradually become ineffective, as its determination of service appropriateness diverges from reality.

To an extent, any current SDM registry is designed to handle certain types of change

autonomously and dynamically. For example, the changing availability of services is handled through the leasing of service advertisements. Indeed, if the qualities of an available service change, it would be hoped that the service provider would reflect this by modifying the service description advertisement, thus updating the world model. However, other aspects of a recommending registry conforming to the theoretical design would be defined statically: the appropriateness rules, proportional criteria weightings, and elements of the world model that were not expected to change on a regular basis. These could only be updated through manual intervention.

Consider the following examples. In the university department which Alice is currently visiting, certain corridors have been closed for a three month refurbishment. To reflect this, the architectural information in the department registry world model would need to be modified, otherwise distances to printers calculated using the physical location rule would be inaccurate, and inappropriate printers might be ranked highly in Alice's recommendation. Alternatively, imagine that some of the DNA sequence-analysers on the grid began offering a facility which enabled a consumer to assess the current status of a running analysis, and to modify some of the analysis parameters dynamically. Over time, it becomes clear that researchers value this facility, so it would seem valid to take this new criterion into consideration when determining the appropriateness of an analyser service. Consequently, the developer of a grid recommending registry would need to introduce a corresponding unconditional rule which assigns greater appropriateness to a service with this new facility. Moreover, the proportional criteria weighting would need to be modified to reflect the importance of this newly introduced criterion. Only through this adaptation could appropriate analysers continue to be highly ranked in a recommendation generated for Bob.

The need for manual modifications to a deployed recommending registry, in order to reflect relevant changes, places a significant burden on the developer or other responsible party. If the developer does not continue to make necessary modifications to a registry over its lifetime, the effectiveness of the recommendations generated could gradually deteriorate.

Inaccuracy in the World Model

With the theoretical design, the world model plays an integral role in a recommending registry. The various criteria rules that determine service appropriateness operate by assessing the current state of the world as presented by the model attributes. To enable

effective recommendations to be generated, a world model would thus need to provide an accurate reflection of the elements that it aims to represent: the requesting consumer, the available services, and other relevant aspects of the computing and physical world. However, there are several reasons why aspects of a registry model might be inaccurate.

On a mundane level, mistakes could always be made in the acquisition and representation of world model attributes. A faulty sensor might record an incorrect value in tracking a requesting consumer's location, or a human service provider might accidentally advertise a service with the wrong attribute value. Alternatively, as was discussed in the previous point, if relevant changes occurred in the real world and affected model attributes were not consequently updated, then the model would no longer accurately reflect the current state of things. On a more abstract level, a source of model information might perceive world aspects from a certain viewpoint, which most others would disagree with. For example, a service provider might have a rose-tinted perception of his service, which would be reflected in the advertised service description attributes. Those who had actually used the service might consider the description inaccurate.

There is also the interesting issue of whether a source of information would always wish to provide accurate attributes for a registry world model. A requesting consumer would be likely to provide contextual information that accurately reflected his current state, being motivated to obtain an effective recommendation of services appropriate for his particular circumstances. A registry developer should similarly be motivated to generate effective recommendations, and would therefore probably attempt to ensure that aspects of the world model defined by him were accurate. However, from the perspective of service providers, it might actually be advantageous to them to supply *inaccurate* service descriptions for a registry world model.

A provider is motivated to advertise a service within a registry to gain users. In a competitive service market, more users might equate to more money, either through fees paid by the users themselves or through some form of advertising. It therefore seems likely that some providers might be somewhat economical with the truth when describing services, in an attempt to attract greater custom than they perhaps deserve. For example, the provider of a very slow DNA sequence-analyser might lie in the advertisement submitted to a grid recommending registry, claiming that the service could provide a massively high processing-throughput level. The service would consequently be highly ranked in generated recommendations, with the result that Bob and many others might select it.

The provider would then benefit from their paid fees. Unfortunately, from the perspective of the users, the service would prove exceedingly inappropriate, with the analysis process taking weeks to complete rather than the expected few days. In a more malicious instance, the service provider might simply register the fee, and not perform any analysis at all!

Given the reasons discussed above, there would seem to be a distinct possibility that the world model of a registry which conformed to the theoretical design could be inaccurate in some way. This obviously diminishes the likelihood that such a registry could generate effective personalised service recommendations. The appropriateness rules and proportional criteria weightings defined by a developer might theoretically be well-defined, but if the world model information over which they operate was inaccurate, then determined service appropriateness would be of little value.

5.2.2 Considering a Generic Mechanism

What would a generic mechanism, developed to support this theoretical design consist of? To what extent could it simplify the construction of a recommending registry?

The theoretical design requires the developer to construct a recommending registry that comprises a world model, appropriateness rules and proportional criteria weightings. A supporting generic mechanism would probably consist of a "skeleton" implementation of such a designed registry. At its core, it might provide a basic service discovery mechanism, which handled the advertisement and leasing of provider-supplied service descriptions. On top of this, it might provide a framework into which world model content, rules and weightings, specific to the particular scenario, could be "plugged in" by the developer. A skeleton world model might also be provided, which adopted a standardised way of defining and manipulating contextual information. Support might further be provided for the acquisition and processing of contextual information through an infrastructure such as Dey's Context Toolkit [30]. This should all theoretically simplify the developer's task of developing a working world model, and of devising the appropriateness rules that access the model. The skeleton implementation might also provide support for calculating multi-criteria service appropriateness, by combining values generated by developer-specified appropriateness rules according to a developer-specified proportional criteria weighting. It might also handle the generation and return of personalised service recommendations, ordering services into a ranked list by their overall appropriateness values.

Clearly, a generic skeleton registry thus described could provide a solid technical base

on which to construct a working recommender, simplifying aspects of the developer's task. However, such a generic mechanism would be unable to provide any specific support as to the particular world model content, appropriateness rules and proportional criteria weightings which a developer himself would need to define and "plug" into a skeleton registry in order to tailor it to a particular deployment scenario. The design and implementation of such components could potentially be a very challenging and time-consuming process, with much bespoke, intricate code needing to be written. Thus, even with the aid of such a supporting generic mechanism, a developer would be likely to face a particularly complex task when constructing a working recommending registry that could generate effective personalised service recommendations.

5.3 Conclusion

In theory, a recommending registry that adhered to this design might well generate effective personalised service recommendations. In practice, however, the likelihood of this occurring is severely reduced. This is because a successful manifestation of the theoretical design is dependent on the near-perfect fulfilment of several disparate conditions; a situation which cannot be guaranteed. Specifically:

- The registry developer must be adept and aware enough so as to define successfully a world model, appropriateness rules and proportional criteria weightings, all together, for the particular deployment scenario and service types involved.
- A deployed recommending registry must be constantly monitored and modified so as to reflect relevant changes that affect service appropriateness determination, in order to avoid a deterioration in recommendation effectiveness.
- World model content obtained from all information sources must consistently and accurately reflect the real state of the world ("Garbage in, garbage out").

Moreover, the style of this theoretical design means that a supporting generic mechanism would be able to do no more than provide basic technical aid in the construction of a recommending registry. The developer would still have to confront the potentially complex and challenging task of defining all the components necessary for determining service appropriateness in the specific scenario.

In conclusion, although this design for a recommending registry is theoretically feasible, it has too many potential drawbacks associated with it to make it entirely satisfactory. The complexity inherent in the design's implementation and deployment, with the consequent likelihood of error, would seem to negate any positive aspects of the design itself. In view of these points, the theoretical recommending registry design would not seem to merit further, more detailed investigation.

5.4 Summary

In this chapter, a theoretical general design for a recommending registry has been presented. This design has been derived primarily from techniques used in the SDM-style recommending systems discussed in the last chapter. The component parts of the theoretical design, namely a world model, conditional and unconditional service appropriateness rules, and proportional criteria weightings, have all been defined and discussed. An explanation has then been given of how recommending registries adhering to this theoretical design would operate, using clarifying examples.

The design's suitability has subsequently been assessed, in terms of its ability to generate effective personalised service recommendations and the potential simplification of recommending registry construction. The positive aspects of the theoretical design have been acknowledged, but three highly problematic features have also been identified and discussed. These, together with the complexity inherent in the design's implementation and deployment would seem to diminish the theoretical design's suitability.

In consequence, the theoretical design cannot be considered suitable as a general design for an effective recommending registry, and there is thus no merit in investigating the design further.

Given this conclusion, an alternative approach to recommending registry design is required. Such an approach is proposed and considered in the next chapter.

Chapter 6

A Novel General Approach Based on Collaborative Filtering

A novel general approach to registry-generated personalised service recommendation is proposed and defined in this chapter. The potential advantages of this new approach are outlined, and the research issues associated with the approach are then identified and discussed. Finally, the association of the approach with Collaborative Filtering is highlighted.

6.1 A Novel Approach for a Novel Design

Given the perceived unsuitability of the theoretical design as a general design for a recommending registry, how can a better one be developed? Essentially, the approach taken by the theoretical design, through the use of a world model, appropriateness rules and proportional criteria weightings, is to approximate how an informed human-being might himself decide which of the available services matching the required type is most appropriate[1]. As an alternative approach, rather than approximate, why not *harness the real decision-making ability of real people directly*? Why not base recommendations on people's actual service selections?

Although allusion has been made in a few research papers on service discovery [5,44,94] to the possible value of past service usage, no-one has yet pursued the notion further.

[1]See decision theory [99,101] for the design's correspondence to human decision-making; the combination of appropriateness rules and a proportional criteria weighting used to determine service appropriateness is essentially equivalent to a weighted multi-attribute utility function.

In contrast, my own research is focused specifically on the concept of how past service selections/usage can be used to determine effective personalised service recommendations for people in the present. The concept is developed into a more precise approach to registry-generated personalised service recommendation in this chapter. The remainder of this thesis is then concerned with how this novel approach was developed into an advanced general design for a recommending registry.

6.1.1 The Novel Approach Defined

In any scenario in which services are available, various people are likely to have made use of them. With every service selection, someone has made a choice, showing a preference for one particular service over all others. Yet why did that person select and use that one particular service? Presumably, it was because he judged it to be the most appropriate service of those available, having taken into account the relevant aspects of his own current situation.

If someone has frequently used a particular type of service, then he should be aware of the important criteria by which to judge service appropriateness, and of the qualities of particular services. He should be aware of important determining factors, such as which services had proved reliable and trustworthy in the past, and which services he is precluded from using because of physical constraints, organisational culture, rules, regulations, etc. Thus, such an experienced individual should make appropriate service selections.

Someone who had not used a particular type of service would initially find it problematic to make an appropriate selection but, over time, would inevitably learn more about service appropriateness and the various services available. He would gain experience through any initial selections of inappropriate services, and could glean useful information by asking others for advice. With this acquired knowledge and understanding, such an individual would also become able to select appropriate services.

Two assumptions are therefore made about a scenario in which services are available:

- An individual selected a service in the belief that it was the most appropriate service of the required type, given his particular situation.
- A significant proportion of past services chosen *were* truly appropriate for the selecting individuals.

If these two assumptions hold true, why not take advantage of these appropriate service

selections? If an uninformed consumer in a particular situation makes a request for a certain type of service, why not base the recommendation on past service selections made by people in a similar situation who had required the same type of service?

More precisely, if the two assumptions hold, I propose that a recommending registry be designed to generate personalised service recommendations according to the following general approach; as in the previous chapter, the abstract model of a recommending registry defined in Section 4.4 will serve as a basis:

- Before and during recommending registry deployment:
  - All service selections in a scenario are recorded, so that an historical sequence of selections is built up over time. Associated with each service selection is the type of service, and a number of "situation attributes". These attributes record those aspects of the situation that an individual seems most likely to have taken into consideration when deciding upon, and then selecting, the most appropriate of the available type-matching services. Clearly, the set of situation attributes recorded is specific to the type of service involved. Those aspects of a situation that are of relevance in assessing service appropriateness could differ significantly from one type of service to another.
- On generating a personalised service recommendation:
  1. As required by the abstract model, a requesting consumer submits a service request to the registry, stating the required type of service and any specific service attributes. The registry responds by acquiring the type-specific situation attributes that define the requesting consumer's current situation. It also identifies those available services with the required type by assessing the advertised service descriptions.
  2. The registry identifies those service selections in the recorded history that were recently made in a situation similar to that of the requesting consumer, and that refer to a service of the required type. Recent service selections are identified by filtering out only those that were made within a recent time-window, such as the last n days. Situation-similar service selections are then identified by comparing each recent service selection's situation attributes against those of the requesting consumer. For brevity, these recent, situation-similar, type-matching service selections will be referred to as "relevant" service selections.

3. The registry uses the relevant service selections to rank the available type-matching services by collective perceived appropriateness for the requesting consumer's current situation. By comparing the selections of each service relative to one another, the registry ascertains each service's ranking in an ordered list. If a service has been selected multiple times by multiple individuals, it would seem likely that this particular service is collectively considered highly appropriate, and should consequently be ranked highly. In contrast, a service that has not been selected at all would seem to be considered highly inappropriate, and should thus be given the lowest ranking possible.
4. The registry returns the ordered list of services to the requesting consumer as a personalised service recommendation. As required by the abstract model, the recommendation is filterable to show only those services that also match the consumer-specified service attributes.

Such a style of recommendation can be intuitively explained to the requesting consumer as "individuals who recently required the same type of service as you, and in a similar situation, were of the collective opinion that the following highly ranked services were most appropriate".

6.1.2 The Approach in Action

For clarification, imagine that the recommending registries in the running examples of Alice and Bob implement this proposed approach. As will be seen, the situation attributes and time-window used to generate a personalised service recommendation obviously depend on the particular scenario and service type involved.

The recommending registry of the university department where Alice gives her presentation records all service selections made within the department, together with corresponding situation attributes. It therefore records the service selection behaviour of academics, researchers, postgraduate students, undergraduate students, and visitors. Many of these people will be able to select appropriate services, given that a significant proportion of them either work in the department, or visit frequently, and use required types of service on a regular basis.

Alice requests a printer recommendation from the registry by submitting a service request. The registry responds by capturing Alice's situation in terms of the printer-specific situation attributes: location, role and time. It is expected that these aspects of

a situation are the most likely to be taken into consideration when departmental printer appropriateness is assessed (e.g. is the printer physically near to my location? is it accessible given my role? would it be busy at this time of day?). The registry also identifies all available printers by assessing the advertised service descriptions.

Next, the registry identifies all those printer selections in the recorded history that were recently made in a situation similar to that of Alice. More precisely, the registry identifies all printer selections that were made in the last x weeks by various individuals with a similar role, at a similar time of day, and in a similar location. The printer selections thus identified could include, for example, those of people who had also recently given presentations in the departmental meeting room, and those of postgraduate students in nearby offices, as such students are also constrained to using publicly-accessible printers.

The registry then assesses these relevant service selections in order to rank the available printers by collective perceived appropriateness for Alice's situation. The resultant ordered list is returned to Alice as a personalised service recommendation. Ideally, truly appropriate printers would be highly ranked, because various people in a similar situation had selected them in the belief that they were physically near, publicly accessible, had low usage at this time of day, and could print documents of good quality, both reliably and quickly.

In the case of Bob, the recommending registry to which he subscribes records all service selections made within the grid in which it is deployed, together with corresponding situation attributes. It therefore records the service selection behaviour of university researchers, industry researchers, and anyone else actively making use of grid resources. Many of these people will be able to select appropriate services, given that a significant proportion of them frequently run computational experiments that require certain types of grid service.

Bob requests a DNA sequence-analyser recommendation from the registry by submitting a service request. The registry responds by capturing Bob's situation in terms of network location, the analyser-specific situation attribute. It is expected that this aspect of a situation is the one most likely to be taken into consideration when analyser appropriateness is assessed (e.g. does the analyser have good responsiveness and performance from my network location?). The registry also identifies all available analysers by assessing the advertised service descriptions.

Next, the registry identifies all those analyser selections in the recorded history that

were made in the last y weeks at a network location similar to that of Bob. The recent analyser selections of other researchers in Bob's own research group might be identified, for example.

The registry then assesses these relevant service selections in order to rank the available analysers by collective perceived appropriateness for Bob's situation. The resultant ordered list is returned to Bob as a personalised service recommendation. Ideally, truly appropriate analysers would be highly ranked, because various people in a similar situation had selected them in the belief that they were inexpensive, responsive, and could process genome data quickly, accurately and reliably.

6.1.3 Potential Advantages of the New Approach

My proposed new approach has so far been described at an abstract level only. However, it is already evident that a general design for a recommending registry based on this approach should have several advantages, particularly over the theoretical design developed in the previous chapter, making further development justified. The main advantages are:

Simplicity Firstly, a design based on the proposed approach should not have the complexity of the theoretical design, making the implementation and deployment of a recommending registry a simpler proposition. Clearly, a registry developer would need to decide upon the situation attributes and time-window of each service type, and how to record service selections. However, he would not need to develop the complex world model, appropriateness rules and proportional criteria rules of the theoretical design, thus obviating the need to address the associated challenging problems. For example, the developer would not need to address the complexities of world modelling, such as how to measure or represent a service's quality or reputation. In terms of registry deployment, the developer would not face the prospect of needing to monitor and modify the model, rules and weightings regularly in order to ensure that relevant change was reflected.

In view of the above, a well thought out general design based on the proposed approach should allow an associated, supporting generic mechanism to be constructed, which could provide a more complete implementation of a recommending registry than that associated with the theoretical design (see Section 5.2.2).

Automatic Reflection of Change Secondly, a recommending registry designed according to the proposed approach should reflect change automatically. People should

naturally factor important change into their decision-making, and since service appropriateness is determined by people's recent service choices, generated recommendations should also reflect this change.

Consistently Effective Recommendations Thirdly, a design based on the proposed approach should enable a recommending registry to generate effective service recommendations consistently. With the theoretical design, the generation of effective recommendations is mainly dependent on the personal ability of the registry developer to define successfully the process for determining service appropriateness. In contrast, with this new approach, recommendations would be based on the collective decision-making ability of multiple people who had actively selected and used a scenario's services. Given that many of these people would have the ability and personal motivation to select truly appropriate services, the resultant recommendations should correspondingly be effective on a consistent basis. Interestingly, the process of determining service appropriateness in terms of the collective opinion of a group of people shares some similarity with an idea that has recently been put forward in the general media. In a recently-published book, "The Wisdom of Crowds" [108], Surowiecki draws upon academic research and real-world examples to argue that large groups of people are consistently "smarter" than an expert in making a correct decision.

Harder to Influence Recommendations Fourthly, the proposed approach should make it harder for an individual service provider to influence a recommending registry. With the theoretical design, a provider could easily ensure that his service was highly ranked in a recommendation by simply lying about its qualities in the advertised description submitted to a registry. In contrast, when service appropriateness, and corresponding recommendation ranking, is determined purely by collective service selection behaviour, an inaccurate description should have little impact.

The main disadvantage of the proposed new approach would seem to be the general issue of privacy, in this case associated with the recording of people's service selection behaviour for the service selection history. Although this issue would need to be considered at a later date, it is not of the magnitude of the disadvantages associated with the theoretical design of the previous chapter, and as such was not considered a hindrance to further investigation of the new approach.

6.2 From Abstract Approach Towards Advanced Design

6.2.1 Research Issues

In order for the abstract approach to be developed into a general design, the following research issues needed to be considered.

Recording Service Selections How are service selections and their associated situation attributes recorded? A basic approach would be to require each person who made a service selection to specify details manually. However, this approach would probably be unpopular, as it would require people to expend much time and effort. Submitted data might frequently be inaccurate, owing to unintentional human error or intentional lying. A more unobtrusive approach would be to record details of service selections automatically, as they occurred. This would minimise human involvement, but would obviously require some form of underlying distributed mechanism to capture both service selections and associated situation attributes.

Acquiring a Requesting Consumer's Situation Attributes How are the situation attributes that define a requesting consumer's situation acquired? Like service selection recording, the situation attributes could either be specified manually by the consumer, or acquired automatically.

Choosing Situation Attributes How is a decision made as to which set of situation attributes is recorded along with a service selection? As was noted earlier, the attributes chosen should record those aspects of a situation that are most relevant in the assessment of service appropriateness for a particular service type. This is because an individual's specific state in terms of these aspects will greatly affect how he perceives the appropriateness of type-matching services, and his consequent service selection. If the correct set of situation attributes is recorded, then many of the situation-similar service selections that the registry identifies should be appropriate selections for the requesting consumer himself, enabling an effective recommendation to be generated. If incorrect situation attributes are recorded, then inappropriate service selections will be identified, and an ineffective recommendation generated.

Choosing a Length of Time-Window How is a decision made as to the length of time-window used by a registry to identify the recent relevant service selections? The motivation for using time-windows, one for each service type, is to enable a registry to respond quickly to any changes which affect service appropriateness. People should factor change into their service selections. Thus, if important changes occur which cause people to modify their selection behaviour, a shorter time-window should enable change to be reflected more rapidly in the generated recommendations. However, the shorter the time-window, the smaller the number of service selections on which a recommendation would be based. With less information, a registry might generate less effective recommendations.

Identifying Situation-Similar Service Selections How should situation-similarity be defined in order to identify relevant service selections? As was noted earlier, the assessment of situation-similarity would involve comparing a service selection's situation attributes with those of the requesting consumer. Clearly, if the attributes match exactly, the service selection was made in an identical situation, and is therefore relevant. However, what of service selections made in "non-identical" situations to the requesting consumer, with attributes that do not match exactly? Some of these other situations could be considered similar to the identical consumer's situation, in terms of similarly perceived service appropriateness. Thus, many of the service selections made in these situations should also be appropriate selections for the requesting consumer, and would therefore be of relevance in generating the recommendation. To enable the identification of these other relevant service selections, how could such similar, but non-identical, situations be determined?

Generating a Recommendation How is a recommendation generated? Using the relevant service selections identified, the registry must rank the available type-matching services by collective perceived appropriateness. Various factors might be considered in deducing whether one service was perceived as being more appropriate than another: the number of individuals who had selected each service, the number of times each service was selected, or the order in which services were selected. How might a service provider attempt to influence a recommending registry in order to achieve a high recommendation ranking for his service? How might such devious attempts be combated?

Evaluating Recommendation Effectiveness Would a recommending registry that adhered to the developed general design generate effective personalised service recommen-

dations? Solutions might be found to the research issues discussed above, enabling the development of a registry design that, in technical terms, comprehensively defines the process of recommendation generation. However, the design would fulfil the research aim of Section 4.4 only if an adhering registry could generate *effective* recommendations: i.e. the services ranked highly in a recommendation were truly appropriate choices for the requesting consumer. How can such recommendation effectiveness be evaluated?

6.2.2 Research Issues Addressed

I investigated the research issues detailed above within the context of a real-world scenario, and ultimately developed the abstract proposed new approach into a advanced general design for a recommending registry. From a research perspective, the most important issues were those concerned with how a registry generates recommendations from a service selection history, and whether these recommendations are effective. These issues encapsulate the core idea of the new approach and needed to be addressed successfully in order for the viability and validity of the basic concept to be demonstrated. I therefore concentrated primarily on developing general solutions for the last five research issues, which are concerned exclusively with the generation or evaluation of recommendations.

Less detailed attention was given to the first two research issues, which are concerned with how a service selection history is recorded and how a requesting consumer's situation attributes are acquired. While these aspects would be important if a registry were to be deployed in a particular scenario, they were of minor research importance in an exploratory investigation.

In terms of this thesis, therefore, the seven research issues are addressed as follows.

Chapter 7: An evaluation scheme is developed for assessing the effectiveness of recommendations generated by a recommending registry. This scheme is of use in investigating possible solutions to many of the other research issues.

Chapter 8: Basic solutions are given to all six remaining research issues, resulting in a basic general design for a recommending registry. The viability and validity of this basic design is assessed in Chapter 9.

Chapter 10: Further investigation is made into the issue of generating a recommendation from relevant service selections.

Chapter 11: Further investigation is made into the issue of defining service selection situation-similarity.

Chapter 12: Further investigation is made into the two issues of choosing situation attributes and time-windows.

Chapter 13: An advanced general design for a recommending registry is defined.

6.3 Association with Collaborative Filtering

My approach to personalised service recommendation, based on the idea of making use of the decision-making ability of real people, can be seen as a form of automated Collaborative Filtering (CF). CF has been described by two of its early researchers, Riedl and Konstan, as "any mechanism whereby members of a community collaborate to identify what is good and what is bad" [100].

In terms of existing Computing Science research, various computing systems have been developed which use a form of automated CF to recommend appropriate items within particular domains, such as films [55], books [3], newsgroup articles [68] and jokes [46]. Essentially, these "recommender systems" collect individuals' opinions of domain items, and then collate them to generate item recommendations for others. However, with one limited exception, no research has been undertaken into the use of automated CF to recommend services. Moreover, the CF approaches used by most developed recommender systems are completely unsuitable for the style of operation required by a recommending registry, and are very different to my proposed approach.

Most CF recommender systems require a user to express opinions of items in the domain from which a recommendation will be made, typically as explicitly-specified numeric ratings. The system then identifies like-minded individuals, who have similarly rated the considered items. It recommends to the user those items which he has not personally rated, but which the like-minded individuals have collectively rated highly. A user is thus likely to obtain a recommendation of appropriate items that he is unaware of, once he has rated several domain items of which he has an opinion. This form of CF does appear to work effectively in taste-based domains such as books and films.

However, this style of CF approach would clearly be unsuitable for a recommending registry generating personalised service recommendations. An uninformed consumer is

likely to request a recommendation *precisely* because he has no opinion of any of the scenario services available, either through lack of awareness or experience. A recommending registry must therefore be able to recommend appropriate services to the requesting consumer immediately, without his needing to express any form of opinion about the services available, or attempting to develop one by expending time and effort using several inappropriate services. My proposed new approach to personalised service recommendation meets this requirement.

A different CF approach, taken by a few researchers, is to focus on users' sequences of activity. With Chalmer's Path Model [14,15], for example, a user's sequence of activity, or "path", within a domain such as web-pages is recorded. It is assumed that the sequence in which items are chosen is of importance, with each item being related in some way to the items chosen before and after. To generate a recommendation, a user's recent path segment is compared against the activity paths of others, to identify segments where similar behaviour has occurred. Items which have occurred in others' similar path segments, but do not occur in the user's own segment, are then recommended. Chalmer's Recer system [14, 15] uses the Path Model to generate web-page recommendations based on a person's browsing activity. The RECO system [91] also examines a person's sequence of domain activity and those of others, in order to suggest what the person might like to use next.

In a way, this style of CF approach is somewhat similar to my proposed approach, in that peoples' actual behaviour in terms of item choices is used to deduce which items to recommend. However, as with the main CF approach discussed initially, this approach would also be unsuitable for a service recommending registry. A service recommendation could not be generated immediately, but only after the requesting consumer had built up an activity sequence by selecting and using various services. Moreover, if the consumer had made inappropriate choices, then further inappropriate services would probably be recommended to him.

A partial CF approach is used in the PILGRIM mobile recommendation system [11,12], the one recommender system which does generate a form of personalised service recommendation. PILGRIM is designed to operate in an ubicomp environment, and recommends an ordered list of services that are associated with locations physically near to the requesting user. In terms of CF, the system deduces the locations of services by assessing service usage history. Essentially, the system records the particular latitude / longitude location

of anyone who uses a service. For each service, PILGRIM "draws" an ellipsoid through the locations at which the service was used, and then defines the service location as being at the centre of this ellipsoid's mass. The physical distance between a requesting user and the service is calculated in terms of a distance metric based on the ellipsoid, using the user's submitted location and the service's deduced location. However, apart from the fact that service locations and distance metrics are deduced through a form of CF, rather than being explicitly specified by the providers, PILGRIM is conceptually no different to the Websigns system [96] discussed earlier. In generating personalised service recommendations, both determine service appropriateness only in terms of the single criterion of physical location.

Despite the obvious differences between my proposed approach to personalised service recommendation and the Collaborative Filtering approaches to item recommendation described above, the underlying idea of using people's past behaviour would seem to imply commonality. In view of this, I will refer to my proposed approach to personalised service recommendation as a CF-based approach, and the subsequent general design for a recommending registry as a CF-based design.

6.4 Summary

In this chapter, my novel approach to registry-generated personalised service recommendation has been proposed and defined. This approach is based on the concept of harnessing the real decision-making ability of real people directly, through analysis of their past service selections. The approach has been illustrated through the running examples of Alice and Bob in their respective scenarios.

Development of the new approach into a general design for a recommending registry has been justified through the enumeration of its potential advantages, and the research issues which need to be addressed in such development have subsequently been defined and discussed. Finally, the association of my approach with Collaborative Filtering (CF) has been highlighted, and its differences with existing CF approaches noted.

One of the research issues identified, that of evaluating recommending registry effectiveness, is addressed in the next chapter.

Chapter 7

Evaluating Recommending Registry Effectiveness

In this chapter, my evaluation scheme for assessing the effectiveness of a recommending registry that adheres to the CF-based design is motivated, contextualised and defined.

7.1 The Importance of Recommendation Effectiveness

The basic aim in providing registry-generated personalised service recommendations is to ease the burden of service selection for uninformed consumers. The recommending registry itself would undertake the process of identifying available type-matching services and assessing their appropriateness, and the resultant generated recommendations should ideally rank truly appropriate services highly. A recommendation will be effective if a consumer is indeed able to select an ideal service with minimum time and effort, from the top end of the ordered list. The effectiveness of a recommending registry can thus be determined through its ability to generate effective recommendations. How to assess such effectiveness is therefore of the utmost research importance.

In view of this, I devised an original scheme for evaluating the effectiveness of a recommending registry which conforms to my CF-based approach. The evaluation scheme was used to assess the CF-based design in all stages of its development in this research, and could be used in the assessment of any CF-based design adhering recommending registry in any deployment scenario.

7.2 The Inapplicability of Existing Evaluation Schemes

Although some of the previously discussed systems that generate forms of service recommendation have been evaluated in various ways, none of the schemes used are suitable for assessing the effectiveness of a design-adhering recommending registry. In certain cases, such as INS [1] and Context Attributes [72], only the system's technical performance has been evaluated, with the effectiveness of generated recommendations not being assessed at all.

In contrast, both the Personal Router [73] and PILGRIM [12] systems have been evaluated in terms of whether their generated recommendations would have enabled a user to find an appropriate service. For example, the PILGRIM system was evaluated through a simulation, with recommendations being generated for "users" at random locations in an imaginary area, which had been seeded with imaginary past service uses. Each recommendation was evaluated in terms of its ranking of the physically nearest service, which was considered most appropriate, and the results were aggregated to provide an overall assessment of system effectiveness from the user's perspective. Clearly, this evaluation scheme would seem to fit well with the concept of recommending registry effectiveness. However, the scheme obviously cannot be used to evaluate a design-adhering recommending registry, being specifically designed to evaluate PILGRIM, or a similar style of recommender system that determines service appropriateness in terms of physical location. A design-adhering registry would operate in a completely different way, and the appropriateness of recommended services would depend on the particular deployment scenario and service type involved. The Personal Router evaluation scheme is also unsuitable for similar reasons.

In addition, the evaluation schemes used within the general area of CF are of no relevance. Such specialised schemes, used to evaluate item recommender effectiveness, are not suitable for evaluating a recommending registry which operates according to a CF approach different to the existing ones.

7.3 Inspiration from Information Retrieval

7.3.1 General Lessons

If none of the evaluation schemes for the previously discussed systems are applicable, can inspiration be found in another area of Computing Science research? The area that I identified as being of most relevance is that of Information Retrieval (IR) [115]. I realised that

a proposed recommending registry and an IR system have distinct similarities: the aim of a proposed recommending registry is to recommend appropriate services available in a scenario to a user who makes a service request, whilst an IR system aims to recommend relevant documents contained in a document collection to a user who makes an information query. Like a personalised service recommendation, the document recommendation returned to a user will generally be in the form of an ordered list, with documents being ranked by perceived relevance.

Researchers in IR also have a perception of IR system effectiveness that is similar to my perception of recommending registry effectiveness. Consequently, IR evaluation schemes provide a useful indication of how registry effectiveness could itself be assessed. An IR system is considered effective if it generates effective document recommendations, which rank truly relevant documents highly. Thus, the overall effectiveness of an IR system operating over a particular document collection is generally determined by assessing the effectiveness of document recommendations generated in response to representative information queries. A similar approach could be taken when assessing a design-adhering recommending registry deployed in a particular scenario. Overall effectiveness could be determined by assessing the service recommendations generated in response to representative service requests made in different situations.

Many IR evaluation schemes are also designed to be generally applicable, enabling them to be used in the assessment of any IR system, regardless of system implementation or document collection involved. As such, the key assumption made by these schemes in evaluating IR system effectiveness is that a generated document recommendation takes the form of an ordered list. The scheme used to evaluate a design-adhering recommending registry should also operate in a similar way. Clearly, an adhering registry would behave according to the general CF-based design, but no other assumptions could be made about the particular deployment scenario or the service types involved. Thus, the evaluation scheme must also determine registry effectiveness on the single assumption that a generated service recommendation takes the form of an ordered list, as specified in the design.

7.3.2 Cooper's Perspective

In terms of specific IR evaluation schemes, the commonly used Precision-Recall method [115] is not suitable for use in the evaluation of a design-adhering recommending registry as it is too focused on document retrieval. However, after investigation, I discovered a

lesser known scheme developed by William S. Cooper [24] in 1968, and this is relevant.

Cooper [24] stated that "the primary function of a retrieval system is conceived to be that of saving its users, to as great an extent as is possible, the labor of perusing and discarding irrelevant documents in their search for relevant ones". He assumed that, when faced with an information requirement, a user would search through a document collection one element at a time, until he either found enough relevant documents that satisfied his requirement, or gave up. Thus, Cooper considered that the purpose of an IR system was to "optimise" the user's search order, through the use of a document recommendation. More precisely, when a user submitted an information query, the IR system would return the collection documents in the order in which it considered the user should undertake his search, so as to satisfy his information requirement with the minimum time and effort. Following the order suggested by the generated recommendation, the user would first search the documents with the highest rank, then those with the next highest, and so forth. The smaller the number of irrelevant documents that the user needed to peruse and discard in his search for relevant ones, the less time and effort required, and the more effective the recommendation.

Based on this notion of recommendation effectiveness, Cooper devised an evaluation scheme for assessing the effectiveness of an IR system. At its core was the notion of the "Expected Search Length" (ESL) of a recommendation. This value referred to the number of irrelevant documents that a user would need to assess and discard in searching for a required number of relevant documents, if he followed the recommended search order. The smaller the Expected Search Length, the more effective the recommendation. Thus, to determine the overall effectiveness of an IR system operating over a particular document collection, the ESLs of recommendations generated in response to representative information queries were calculated, and the results aggregated.

The function of a recommending registry is similar to that of an IR system, as perceived by Cooper. To paraphrase Cooper, *the primary function of a recommending registry is conceived to be that of saving its users, to as great an extent as is possible, the labour of perusing and discarding inappropriate services in their search for an appropriate one.* When an uninformed consumer needs a service of a particular type, he submits a service request to the recommending registry. The registry responds by first identifying those type-matching services available in the scenario, and then returns them as an ordered list, ranked by perceived appropriateness. Essentially, through the use of this personalised service

recommendation, a recommending registry is suggesting the order in which it considers the consumer should search through the available type-matching services, so as to find an appropriate service with the minimum time and effort. The uninformed consumer could follow this suggested search order, assessing the services with the highest rank, then those with the next highest, until he either found an appropriate service, or gave up. The smaller the number of inappropriate services that the consumer needed to assess in his search for an appropriate one, the less time and effort required, and the more effective the recommendation. The most effective recommendation would therefore be one in which the highest-ranked, and first assessed, service was an appropriate choice for the requesting consumer.

If the notion of service recommendation effectiveness can be seen as similar to Cooper's notion of document recommendation effectiveness, it follows that Cooper's style of evaluation scheme is likely to be applicable for the assessment of a design-adhering recommending registry. A service recommendation should be assessed in terms of how much of the ordered service list a consumer needs to consider before he finds an appropriate service. To determine the overall effectiveness of a recommending registry deployed in a particular scenario, two steps need to be taken. Firstly, recommendations generated in response to representative service requests made in different situations need to be assessed as outlined above. Secondly, the results of such an assessment need to be aggregated.

7.4 The Evaluation Scheme

The evaluation scheme I devised to assess the effectiveness of a design-adhering recommending registry is defined below. The scheme is a more comprehensive version of the approach briefly outlined in the paragraph above, at the end of Section 7.3.2. As such, it takes its conceptual inspiration from Cooper's evaluation scheme for assessing IR system effectiveness, but has been designed from first principles so as to take into account all the specific requirements associated with the assessment of recommending registry effectiveness.

7.4.1 What are Representative Service Requests and Appropriate Services?

If a design-adhering recommending registry deployed in a particular scenario is to be evaluated according to the approach briefly outlined above, two basic issues must first be addressed. Firstly, what representative service requests are used, and how are they chosen? The set of service requests used is very important, as registry effectiveness will be evaluated in terms of the effectiveness of the service recommendations that are generated in response. Secondly, for a representative service request, which of the available type-matching services are appropriate choices, and how are they identified? Clearly, only if appropriate choices are known can the corresponding recommendation be assessed in terms of how much of the ordered list needs to be considered before an appropriate service is found.

Similar issues have been faced in evaluating the effectiveness of IR systems, in terms of choosing representative information queries and identifying the associated relevant documents. In certain cases, the evaluators have themselves defined queries that they considered a typical user would be likely to make [35]. In others, a representative selection of queries has been identified by mining the recorded log of those submitted to a similar IR system in the past [116]. For each of the queries chosen, human "assessors" have then manually searched through the document collection over which the evaluated IR system was operating, marking those documents which they considered relevant. If the system was operating over a document collection concerned with a specialised domain of knowledge, such as medicine or law, then domain experts might be used to identify the relevant documents for each query. With these representative queries and associated relevant documents, it has then been possible to assess IR system effectiveness using one of the IR evaluation schemes. It should be noted that Cooper's scheme does not specify how such queries and associated documents should be identified, being purely concerned with assessing document recommendation effectiveness.

A design-adhering recommending registry deployed in a particular scenario would already have access to a recorded log containing potential representative service requests with associated appropriate services: the service selection history. A history entry records a past service selection made, together with the service type, and attributes describing aspects of the selecting individual's situation. Theoretically, rather than the selecting individual, an uninformed consumer could have been in the same situation at the same point

in time, and required the same type of service. He could have submitted a request to the recommending registry for the entry-recorded type of service, specifying his situation as the same set of entry-recorded situation attributes. For evaluation purposes, therefore, a history entry can be used as the basis for a representative service request.

If the service selection recorded in a history entry was made by a knowledgeable and experienced individual, the selected service itself should have been appropriate for a requesting uninformed consumer in the same situation. Thus, such a history entry will contain all the information required both to derive a representative service request, and to evaluate the effectiveness of the recommendation generated in response. A history entry of this type will be referred to as an "experienced service selection entry" (ESSE).

7.4.2 The Sequence of the Evaluation Scheme

The evaluation scheme I devised for assessing a design-adhering recommending registry is based on ESSEs. The use of ESSEs ensures that a registry is evaluated in terms of responses to service requests that would be typical within the deployment scenario. A registry is assessed according to the following sequence of steps:

1. ESSEs are identified in the service selection history.
2. Recommendations are generated for every ESSE-derived service request.
3. Each recommendation is assessed against the corresponding ESSE-recorded appropriate service actually selected by the experienced individual. These results are then aggregated to provide an overall assessment of recommending registry effectiveness.

The three steps are further detailed below.

1. ESSE Identification:

Firstly, the assumption is made that the service selection history used by the registry has been operating for some time, recording all service selections made within the deployment scenario. Next, ESSEs are identified by assessing all those history entries that refer to service selections made within an earlier segment of time. For example, if the day of evaluation was 15th July 2004, all entries that occurred between 14th June 2004 and 14th July 2004 might be assessed. To determine whether an entry is an ESSE, the prior experience of the selecting individual is considered. The individual should have been

experienced enough to choose an appropriate service if he had selected and used that type of service before, on multiple recent occasions. More precisely, if the individual is recorded in the history as having selected and used that type of service at least m times in the n days before the date-time of the considered service selection, then he is considered experienced, and the history entry an ESSE. For the particular scenario and service type involved, the evaluator needs to choose values of m and n that do indeed lead to the identification of history entries which record appropriate service selections made by experienced individuals.

2. Recommendation Generation:

Next, for each past ESSE identified, the recommending registry is used to generate a personalised service recommendation in response to the derived service request. That is, a recommendation is generated as if an uninformed consumer at that point in time had been in the recorded situation and required the recorded type of service. This requires two constraints to be placed on the usage of the service selection history in generating the recommendation.

Firstly, only history from *before* the date-time of the ESSE service selection can be taken into consideration. If a recommendation is being generated as if an uninformed consumer at that point in time had made a service request, then all service selections recorded after that point would be in the future, and would not have happened yet. For example, if a service request was "made" on the 16th June 2004 at 14:32:05, no service selections after that point could be considered, even though the history recorded activity up to the day of evaluation on the 15th July 2004.

Secondly, all service selections made by the experienced individual responsible for the ESSE service selection must *not* be taken into consideration. The experienced individual is being used by the evaluation scheme as an external arbiter, who specifies which service is appropriate for a corresponding service request made by an uninformed consumer at that point in time, in that situation. As such, consideration of his past service selections would not enable a stringent enough evaluation to be undertaken, as it might provide a good indication of what he considered appropriate, unfairly skewing the generated recommendation towards his opinion.

3. Recommendation Assessment and Results Aggregation:

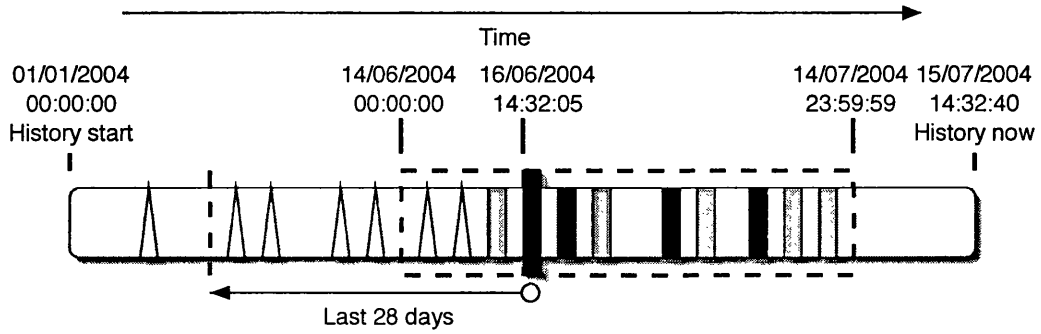
Once a recommendation has been generated for every ESSE-derived service request, each is evaluated against the corresponding appropriate service that was actually chosen by the experienced selecting individual. It is assumed that an uninformed consumer would follow the search order suggested by the recommendation, looking for an appropriate service. It is also assumed that the consumer would find the experienced individual's choice appropriate. Thus, evaluation of recommendation effectiveness is based on the assumption that, in the worst case, the consumer would follow the search order, assessing and discarding every service as inappropriate, until he encountered the experienced individual's appropriate choice. He would then terminate his search, having satisfied his requirement. Clearly, the consumer might well have found an appropriate service earlier in the search order, but the only actual evidence of service appropriateness is the experienced individual's single service selection.

Finally, the individual recommendation effectiveness evaluations are aggregated, to provide an overall assessment of recommending registry effectiveness. A particular method of recommendation evaluation and results aggregation is referred to as an "effectiveness measure". The three interrelated effectiveness measures that I devised for this evaluation scheme - Recommendation Success Probability (RSP), Improvement over Chance (IOC), and Normalised Cumulative IOC (NCIOC) - are detailed in Sections 7.4.3, 7.4.4 and 7.4.5 respectively.

An Example of the Evaluation Scheme in Use

Imagine that the grid recommending registry used by Bob is being evaluated on 15th July 2004. The evaluator decides to assess the registry in terms of ESSEs identified between 14th June 2004 00:00:00 and 14th July 2004 23:59:59 (the night before evaluation day). The fictitious service selection history used by the fictitious grid registry is illustrated at the top of Figure 7.1, as a time-ordered sequence of history entries. The dotted rectangle encompasses the history time-segment from which ESSEs - shown as black vertical bars - are identified. Note that some of the history entries within the time-segment - shown as grey vertical bars - are not ESSEs.

Details of one of the ESSEs identified - shown as a larger black vertical bar - are given below the history illustration. This history entry records a service selection made at 16/06/2004 14:32:05, when User A29 chose DNA sequence analyser B from a network



ESSE:
Service selection: Seq_Analyser B selected by User A29 at 16/06/2004 14:32:05
Situation attributes: (NetworkLocation = 130.209.246 IP subnet)
Number of times Seq_Analyser type used by User A29 in last 28 days: 6 (experienced)

Derived Service Request:
Service type requested: Seq_Analyser
Situation attributes: (NetworkLocation = 130.209.246 IP subnet)

Evaluation Specific Details:
Evaluation service request date-time: 16/06/2004 14:32:05
Selecting individual ignored: User A29

Personalised Service Recommendation (Seq_Analyser):
1. G, D
2. H,
3. *B*, A
4. E
5. W

Figure 7.1: An Example of the ESSE-based Evaluation Scheme

location in the 130.209.246 IP subnet. The entry is identified as an ESSE, as User A29 had selected and used analyser services 6 times in the last 28 days (show as white triangles). This makes him experienced in the eyes of the evaluator, who has defined the minimal experience level for the Seq_Analyser service type to be at least 5 uses (m) in the last 28 days (n).

Once all the ESSEs are identified, the grid recommending registry is used to generate a personalised service recommendation for each of the derived service requests. In the case of the detailed ESSE, a request for a service of type Seq_Analyser is submitted to the registry, with the requesting consumer's situation being specified as the 130.209.246 IP subnet network location. The registry generates the recommendation through assessment of the service selection history, ignoring service selections made by User A29, and those which occurred on or after 16/06/2004 14:32:05.

Finally, each generated service recommendation is evaluated against the corresponding appropriate service that was actually chosen by the experienced selecting individual, as recorded in each ESSE. In the case of the detailed ESSE, User A29 chose Seq_Analyser B, which is ranked in third place in the generated recommendation. The results are then aggregated, to provide an overall assessment of grid recommending registry effectiveness. The three interrelated effectiveness measures of RSP, IOC and NCIOC detailed below would be used.

7.4.3 The Effectiveness Measure of Recommendation Success Probability

Searching Through a Recommendation

As was stated earlier, a personalised service recommendation generated by a recommending registry will take the form of an ordered list, with services being ranked by perceived appropriateness for the requesting consumer (from most to least). An example is shown at the bottom of Figure 7.1. A recommendation could take the form of a *partial* order: that is, the ordered list could contain ties, with multiple services sharing the same rank. This situation will naturally occur if the registry perceives certain services to have the same level of appropriateness. The example recommendation contains two such instances, with services G and D tying for the first rank, and B and A for the third rank. If a generated recommendation contains no ties, with every service distinctly ranked one after the other, it is a *total* order (i.e. a permutation).

As was also stated earlier, it is assumed that a consumer would follow the search order suggested by a recommendation when looking for an appropriate service. The consumer would consider each service in turn, first assessing those with the highest rank, then those with the next highest, until he either found an appropriate service, or gave up. If multiple services tie for the same rank, it is assumed that they are assessed in some arbitrary order. In the Figure 7.1 recommendation, for example, the consumer would either consider D before G, or vice versa. Thus, the process of consumer searching involves a total order being imposed on a recommendation, with any ties being broken through arbitrary ordering of the services involved.

Original Recommendation	**Possible Search Orders**			
1. G, D **2.** H, **3.** *B*, A **4.** E **5.** W	G D H A ***B*** E W	D G H A ***B*** E W	G D H ***B*** A E W	D G H ***B*** A E W

Figure 7.2: Possible Search Orders followed by the Consumer in the Figure 7.1 Example

In the case of a partially-ordered recommendation, the particular total order followed by a consumer will determine the specific number of services that he assesses in his search for an appropriate one. Consider Figure 7.2. This shows the four different total orders in which a consumer could search through the Figure 7.1 example recommendation, which was generated in response to an ESSE-derived service request. In the worst case, it is assumed that the consumer follows his particular search order, assessing and discarding every service as inappropriate, but stopping when he finds B, the ESSE-recorded appropriate choice. In two of the orders, when A is searched before B, the consumer assesses 5 services before stopping. In the other two, when B is searched before A, only 4 are assessed.

In terms of evaluating recommendation effectiveness, a measure could be developed which considered all the different total orders in which a consumer could search through a recommendation, calculating the average number of services that he would be expected

to assess before the ESSE-recorded appropriate choice was found. This would essentially be equivalent to Cooper's concept of Expected Search Length (ESL), but adapted for a service recommendation. In the case of the Figure 7.2 example recommendation, the ESL would be 4.5. If ESLs were calculated for all ESSE recommendations on which an evaluation was based, the different values could then be averaged to provide an overall figure representing recommending registry effectiveness. The smaller the overall ESL, the less time and effort required by the consumer to find an appropriate service using a registry-generated recommendation, and the more effective the registry.

The Concept of Tolerance Levels

Although this measure of overall Expected Search Length would reflect recommending registry effectiveness, it is questionable how meaningful the figure would be when considered from the perspective of actual recommendation usage. A consumer is only likely to follow a recommendation search order so far, before giving up through impatience or frustration if he fails to find an appropriate service. More precisely, if the consumer assesses the maximum number of services that he is willing to consider (defined in this research as his *tolerance level*) without success, he is likely to consider the recommendation a failure, and will try other ways of finding a service. It would therefore seem sensible to consider the effectiveness of a recommending registry from the perspective of the different tolerance levels that consumers might have. That is, if a consumer has a particular tolerance level, what would be the likelihood of his finding an appropriate service using a registry-generated recommendation?

Initially, it might seem that the single figure of overall ESL would enable tolerance levels to be considered. A consumer would appear not to have found an appropriate service if his tolerance level was less than the overall ESL, and to have found one if his tolerance level was more. However, the situation is not as straightforward as this, as the overall ESL would be the *average* of the individual ESLs for all ESSE recommendations. As such, the figure would not reflect how a consumer with a particular tolerance level would have fared in terms of the actual recommendations.

To clarify, imagine a simple evaluation in which a recommending registry has been assessed in terms of six totally-ordered ESSE recommendations, and has an overall ESL of 4. How would a consumer with a tolerance level of 3 have fared? Based on the overall ESL value alone, it might appear that he would never have found an appropriate service,

given that $3 < 4$. However, if the individual recommendation ESLs were <2,2,2,2,8,8> then in 66.67% of cases, the consumer would actually have found one, and after assessing only 2 services! Clearly, the measure of overall Expected Search Length is inadequate as a means of assessing recommending registry effectiveness in the precise manner described.

Recommendation Success Probability Defined

In view of the above, the actual evaluation scheme devised does not make use of the effectiveness measure of Expected Search Length at all. Rather, the effectiveness of each ESSE recommendation is first evaluated in terms of the probability that a consumer with a particular tolerance level would have successfully found the ESSE-recorded appropriate service choice. This probability is referred to as Recommendation Success Probability (RSP), and is calculated according to the function described below.

When calculating the RSP of recommendation r for tolerance level t, where $t > 0$, the following definitions are made:

- a is the rank of recommendation r occupied by the appropriate service.
- $|a|$ is the number of services in rank a.
- s is the number of services in the ranks above a.

Then:

$$RSP(r,t) = \begin{cases} 0 & \text{when } t <= s \\ \frac{t-s}{|a|} & \text{when } s < t < s + |a|, \text{ and } |a| > 1 \\ 1 & \text{when } t >= s + |a| \end{cases} \quad (7.1)$$

To understand the RSP function, it is necessary to consider the search orders that a consumer could follow through a recommendation. In the case of a totally-ordered recommendation with no ties, this is relatively simple, as only the single, explicitly specified search order could be followed. Clearly, a consumer with a tolerance level $t <= s$ would have no chance of finding the appropriate service, as he would not encounter rank a; this is the first case in the function definition. If, however, his tolerance level is greater than s, then he would always find the appropriate service, as he would encounter and assess the single service in rank a. That is, $RSP(r,t) = 1$, where $t > s$. This is simply a rewriting of the third case in the function definition; since $|a|$ must be 1 in a total order, $t > s$ is equivalent to $t >= s + |a|$. Correspondingly, the second case is not of relevance.

For a partially-ordered recommendation, the explanation is slightly more complex, as a consumer could potentially follow any of a number of total search orders, depending on how ties were broken. However, as in the case of a totally-ordered recommendation, the first and third cases in the function definition are fairly self-explanatory. Regardless of the search order followed, a consumer with a tolerance level $t <= s$ would obviously have no chance of finding an appropriate service, as he would not encounter rank a; this is the first function case. Consider Figure 7.2 again, in which a = 3rd rank, $|a| = 2$ (A and B), and $s = 3$ (G, D and H). If $t <= 3$, then the 3rd rank occupied by appropriate service B would never be encountered, for any of the four search orders. In terms of the third function case, a consumer with a tolerance level $t >= s + |a|$ would always find the appropriate service regardless of the search order followed, as he would assess every service in rank a. The appropriate service would be one of these. For example, for any of the Figure 7.2 search orders, B would always be found for a tolerance level $t >= 5$.

In the case of a partially-ordered recommendation in which a is a tied rank, the different potential search orders do have an impact on the calculation of RSP for a tolerance level t, such that $s < t < s + |a|$; this is the second case in the function definition. For such a value of t, a consumer would assess some, but not all, of the services with rank a. Thus, depending on the order in which the rank services were assessed, the appropriate service might or might not be found. For example, in the Figure 7.2 recommendation, if $t = 4$ $(3 < t < 3 + 2)$, then in two of the search orders (when B is assessed before A), B would be found; in the other two (when A is assessed before B), B would not be found.

To calculate the probability that the appropriate service would be found with such a tolerance level, it is therefore necessary to calculate the probability that the appropriate service would be one of the $t - s$ services of rank a assessed by the consumer, regardless of the search order followed. With any search order, it is assumed that the services of rank a have been arbitrarily ordered to break the tie. Imagine that this rank ordering is represented as a service sequence of length $|a|$, as $[service\_one, service\_two, \ldots, service\_|a|]$. Assuming that the consumer assesses the rank services in the sequence specified, then RSP is the probability that the appropriate service would be in one of the first $t - s$ positions in the sequence. Since the sequence is ordered arbitrarily, the probability of the appropriate service being in a particular position is $\frac{1}{|a|}$. The probability of its being in one of the first $t - s$ positions is therefore $\frac{t-s}{|a|}$, as stated in the second case of the RSP function. As a clarifying example, let us calculate the RSP of the Figure 7.2 recommendation r, for $t = 4$.

Reiterating, a = 3rd rank, $|a| = 2$, and $s = 3$. Thus:

$$RSP(r,4) = \frac{4-3}{2} = 0.5 \tag{7.2}$$

This value can be intuitively understood by again considering the four possible search orders illustrated in Figure 7.2: in half of them, a consumer with a tolerance level of 4 would find B.

In terms of the devised evaluation scheme, the effectiveness of a design-adhering recommending registry is based on the measure of overall Recommendation Success Probability for various tolerance levels. More precisely, the ESSE recommendations are first assessed, and the length of the longest recommendation identified. For each ESSE recommendation, the RSP is calculated for every tolerance level between 1 and this longest recommendation length. There is no reason to consider levels beyond this length as, by then, an appropriate service must have been found in all recommendations, since a consumer would have searched through all the available type-matching services. The overall RSP for each considered tolerance level is then calculated by averaging the corresponding RSPs of the individual recommendations. For a particular tolerance level t, where $t > 0$, and set of ESSE recommendations R:

$$Overall\_RSP(R,t) = \frac{1}{|R|}\sum_{r \in R} RSP(r,t) \tag{7.3}$$

The larger the overall RSP for a considered tolerance level, the more effective the recommending registry.

The resulting overall RSP values can then be presented as a graph, by plotting probability against tolerance level. Figure 7.3 shows two example graphs that plot the overall RSP for a ficticious design-adhering recommending registry. In both cases, overall RSP has been presented as a percentage. Overall RSP is simply referred to as "RSP" since, when registry evaluation results are being presented, it is implicit that the results refer to the registry, not to a single recommendation. The graph on the left plots overall RSP from 1 to the largest recommendation length, 68 in this case. Note that the larger the tolerance level, the larger the overall RSP. Thus, the greater the tolerance of a consumer, the greater the likelihood that an appropriate service would be found. As would be expected, overall RSP is 100% at the largest considered tolerance level. The graph on the right plots a "zoomed in" version of the same results, showing the overall RSP of tolerance

levels between 1 and 10. An evaluator should be most interested in the effectiveness of a recommending registry at these early tolerance levels, as it seems unlikely that a consumer would search very far. Overview RSP graphs and zoomed RSP graphs are used to present recommending registry evaluation results later in this thesis.

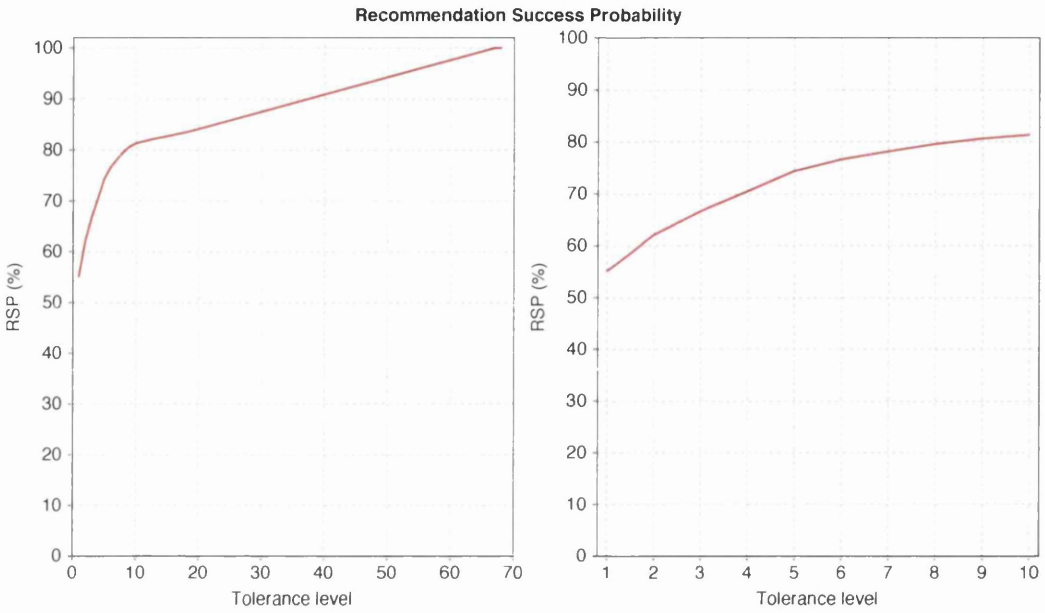


Figure 7.3: Example Graphs Plotting the RSP of a Ficticious Recommending Registry

7.4.4 The Effectiveness Measure of Improvement over Chance

In order to ascertain whether the effectiveness of a recommending registry is a consequence of using the CF-based design, rather than merely being a consequence of chance, a refinement to the effectiveness measure of overall RSP needed to be made. Essentially, the measure needed to be refined so that it could be used to ascertain whether the overall effectiveness of the registry-generated recommendations, which order available type-matching services by design-calculated appropriateness, is better than that of equivalent recommendations which simply order the services arbitrarily, according to chance.

A chance-corrected version of overall RSP has been developed which enables this assessment to be made. Referred to as overall Improvement over Chance (IOC), the measure operates as follows. For each registry-generated ESSE recommendation and considered tolerance level, RSP is calculated as before. However, the RSP of an equivalent recommendation, in which the same type-matching services are ordered arbitrarily, is also calcu-

lated. The RSP of this arbitrarily-ordered recommendation is then subtracted from that of the registry-generated recommendation. This difference in probabilities is referred to as IOC. The overall IOC of the recommending registry for each considered tolerance level is calculated by averaging the corresponding IOCs of the individual ESSE recommendations.

For a particular tolerance level, overall IOC therefore indicates how the overall RSP effectiveness of the registry-generated recommendations compares against that of equivalent arbitrarily-ordered recommendations. If this difference in overall probabilities is positive, i.e. an improvement over chance, then it can be assumed that the effectiveness of the recommending registry is a direct consequence of using the CF-based design. If there is no such positive difference, no such assumption can be made.

Although the measure of overall IOC is a difference of probabilities, in essence it is simply overall RSP in a chance-corrected form. Thus, for a particular tolerance level, the larger the IOC value, the more effective the recommending registry.

In mathematical terms, the IOC of an ESSE recommendation is calculated according to the function described below. For a recommendation r consisting of k available type-matching services, and for a tolerance level t, where $t > 0$, then:

$$IOC(r,t) = \begin{cases} RSP(r,t) - \frac{t}{k} & \text{when } t < k \\ 0 & \text{when } t >= k \end{cases} \tag{7.4}$$

Essentially, the arbitrarily-ordered equivalent of recommendation r can be perceived as a partially-ordered list with one tied rank, which contains all k type-matching services. Thus, its RSP can be calculated using Function 7.1, setting a = 1st (and only) rank, $|a| = k$ and $s = 0$.

To calculate the overall IOC of a recommending registry, for a particular tolerance level t, where $t > 0$, and set of ESSE recommendations R:

$$Overall\_IOC(R,t) = \frac{1}{|R|} \sum_{r \in R} IOC(r,t) \tag{7.5}$$

Like overall RSP, the resulting overall IOC values can be presented as a graph, by plotting probability difference against tolerance level. Figure 7.4 shows an example graph which plots the overall IOC of the fictitious design-adhering recommending registry whose overall RSP was plotted in Figure 7.3. Overall IOC is presented as a difference in percentage points. As in the last figure, overall IOC is simply referred to as "IOC" since, when registry evaluation results are presented, it is implicit that the results refer to the registry, not to a single recommendation.

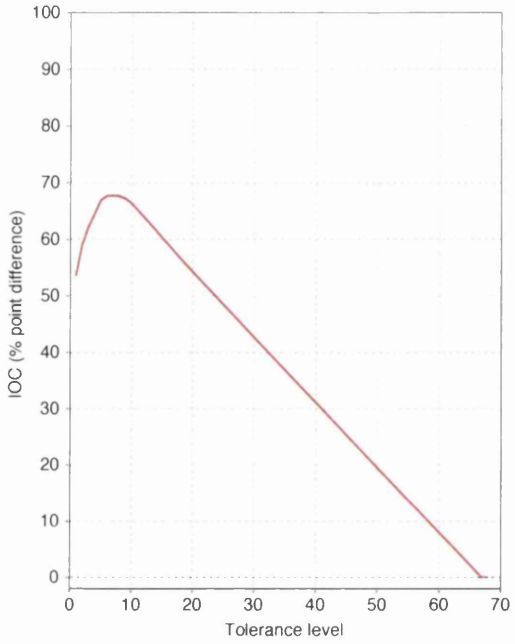


Figure 7.4: An Example Graph Plotting the IOC of a Fictitious Recommending Registry

By comparing Figures 7.3 and 7.4, it is possible to see the impact that chance correction would have on the initial overall RSP results. In the case of this fictitious recommending registry, overall IOC is positive at all tolerance levels but the last, indicating that registry effectiveness is a consequence of using the CF-based design. However, as the tolerance level becomes larger, the probability difference decreases. This is always to be expected as, for an arbitrarily-ordered recommendation, the more services that are assessed, the greater the likelihood of finding an appropriate service. At the largest considered tolerance level, an appropriate service would always have been found, for any recommendation, so there is no difference in the overall RSP of appropriateness-ordered recommendations or arbitrarily-ordered recommendations. IOC graphs are also used to present recommending registry evaluation results later in this thesis.

7.4.5 The Effectiveness Measure of Normalised Cumulative IOC

Although the measure of overall IOC does enable chance-corrected effectiveness to be evaluated, it is not a suitable effectiveness measure to use when investigating the development of the CF-based recommending registry design itself. This is because registry effectiveness is represented as a set of numeric values, one for each tolerance level, which could

make comparison between different registry evaluations rather complicated. The ability to compare one registry evaluation with another is important when assessing competing solutions to the various research issues associated with the development of the registry design. Ideally, to make comparison simpler and quicker, the effectiveness of a recommending registry should be represented by a single numeric value. Moreover, such a value should also indicate how well the registry performs in relation to the "perfect" registry, i.e. one that generates perfect recommendations in which the first ranked service is always appropriate. With such information, it should be possible to identify the best of the competing solutions to each research issue.

The measure of overall IOC was thus further refined to enable registry effectiveness to be represented as such a single numeric value. The refined measure, known as Normalised Cumulative IOC (NCIOC), is defined below. It is assumed that, in evaluating a recommending registry using a set of ESSEs, overall IOC values have been calculated for every considered tolerance level. Then, for a particular tolerance level t, the overall IOC values for levels 1 to t are summed; this sum is referred to as the registry cumulative figure. If the overall IOC values were plotted as a graph, as in Figure 7.4, the figure would represent the area under the curve between 1 and t. Next, the equivalent cumulative figure is calculated as if the recommending registry had actually generated perfect recommendations in response to all ESSE-derived service requests. That is, overall IOC values are calculated for tolerance levels 1 to t, and then summed, as if every registry-generated ESSE recommendation had ranked the ESSE-recorded appropriate service in first place, with all other available type-matching services in ranks below. Finally, the registry cumulative figure is normalised by dividing it by this perfect cumulative figure. The resulting value is referred to as NCIOC.

When calculated for a particular tolerance level t, NCIOC therefore represents the normalised, cumulative, chance-corrected effectiveness of the recommending registry for tolerance levels in the range 1 to t. It seems likely that the tolerance of consumers in searching for an appropriate service will vary. Some consumers will be very impatient, with a low tolerance level, whilst others will be more patient, with a higher tolerance level. Thus, in the single numeric figure of NCIOC, the overall effectiveness of the registry for consumers with tolerance levels in the range 1 to t will be represented. When calculating NCIOC for a particular recommending registry, an evaluator therefore needs to decide upon a value of t which is the largest tolerance level that a typical consumer is likely to

have.

If different solutions to a particular research issue are each tested in the same recommending registry, then it is a simple task to identify the most effective solution by comparing the NCIOC values of the corresponding registry evaluations. The solution responsible for the largest NCIOC value can be considered the most effective. Moreover, the normalised nature of the NCIOC value means that the relationship between the performance of any research solution and that of the maximum possible can be ascertained. In dividing actual cumulative registry effectiveness by that which would be possible if perfect ESSE recommendations had been generated, NCIOC is presented as a value between -1 and 1. An NCIOC value of -1 indicates that the registry generated the most ineffective recommendations theoretically possible; 0 indicates that it generated recommendations that were only as effective as arbitrarily-ordered recommendations; and 1 indicates that it generated the most effective recommendations theoretically possible. If the best of the proposed solutions for a particular research issue only generated a low NCIOC value, such as 0.2, then it would be clear that there was still significant room for improvement: 0.8, to be exact!

The mathematical details of NCIOC calculation are as follows. To reiterate, NCIOC calculation involves the cumulative overall IOC of a recommending registry being divided by the equivalent value that would have been possible if perfect ESSE recommendations had been generated instead. In order to calculate this equivalent value, we must therefore first calculate the IOC of every registry-generated ESSE recommendation as if it were actually a perfect recommendation, which can be done using the following function. For a recommendation r consisting of k available type-matching services, and a tolerance level t, where $t > 0$, then:

$$Perfect\_IOC(r,t) = \begin{cases} 1 - \frac{t}{k} & \text{when } t < k \\ 0 & \text{when } t >= k \end{cases} \tag{7.6}$$

Basically, for a perfect recommendation, RSP can be calculated using Function 7.1, setting $a = 1$, $|a| = 1$, and $s = 0$. With these values substituted, $RSP = 1$ for all $t > 0$, and the IOC function (Function 7.4) simplifies to Function 7.6 above.

Thus, to calculate the NCIOC of a recommending registry, for a particular tolerance level t, where $t > 0$, and set of ESSE recommendations R:

$$NCIOC(R,t) = \frac{\sum_{i=1}^{t}(\frac{1}{|R|}\sum_{r\in R} IOC(r,i))}{\sum_{i=1}^{t}(\frac{1}{|R|}\sum_{r\in R} Perfect\_IOC(r,i))} \tag{7.7}$$

If NCIOC values are calculated for all considered tolerance levels, then the results can be presented as a graph, by plotting NCIOC against tolerance level. Figure 7.5 shows an example graph which plots the NCIOC values of the fictitious design-adhering registry whose RSP and IOC details were plotted in Figures 7.3 and 7.4 respectively. As was discussed earlier, different registry evaluations should be compared in terms of their NCIOC values for a *single* tolerance level t, such as 7. However, such a graph also provides a complete awareness of the various values of NCIOC for different values of t.

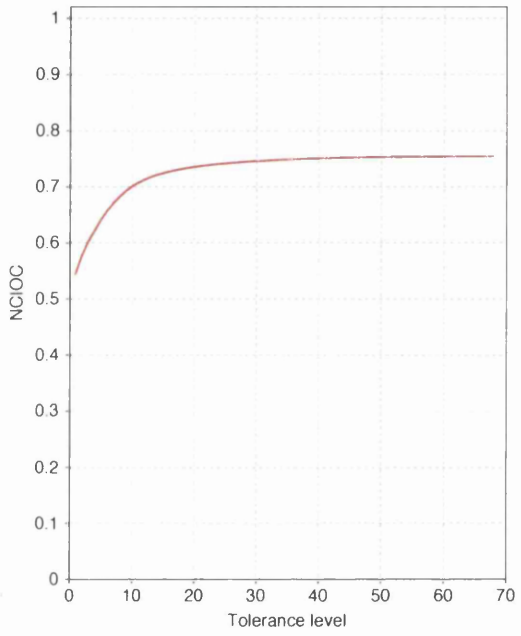


Figure 7.5: An Example Graph Plotting the NCIOC of a Fictitious Recommending Registry

7.5 The Use of the Evaluation Scheme in this Research

The evaluation scheme defined in this chapter, with its identification and use of ESSEs, and the different effectiveness measures of RSP, IOC and NCIOC, is used later in this research to assess the effectiveness of a prototype recommending registry that adheres to the CF-based design. The scheme has been automated, requiring the evaluator to specify only the time segment from which ESSEs will be identified, the criteria for an ESSE, and the tolerance level for comparison of NCIOC values. Various competing solutions to each of the research issues associated with the design are tested in the prototype registry,

and the corresponding evaluation results considered and compared. Evaluation in this research is based on ESSEs relating to only one particular type of service, rather than a variety. However, this approach to evaluation would seem good practice, enabling registry effectiveness to be assessed on a per service type basis. Given the various factors that would affect the generation of recommendations of a particular service type - the choice of situation attributes and time-window length, and the relevant service selections in the service selection history - the effectiveness of a registry could vary between types. Assessing a registry in terms of ESSEs relating to a variety of service types would make it impossible to identify those types that the registry was effective in recommending, and those that it was not.

When evaluation results are presented later in this thesis, the four styles of graph described earlier (shown in Figures 7.3, 7.4 and 7.5) are used, together with a table providing additional details. Figure 7.6 is an example of such a presentation (without the table). Although the graphs all plot registry effectiveness, they each provide different information. The top two graphs plot effectiveness in terms of RSP. This information provides an easily understood presentation of registry effectiveness from the consumer's perspective. For a particular tolerance level that a consumer might have, the probability of his finding an appropriate service can be identified. The bottom left graph plots chance-corrected effectiveness in terms of IOC. By considering whether the IOC values are positive, it can be ascertained whether the registry-generated recommendations were more effective than arbitrarily-ordered recommendations, an improvement over chance. Finally, the bottom right graph plots NCIOC values for different tolerance levels. For a particular tolerance level, an assessment of registry effectiveness relative to perfect registry performance can be made. When registry evaluations for competing research solutions are all plotted on the same graph, the solutions can easily be compared in terms of their respective NCIOC values.

7.6 Summary

The focus of this chapter has been my specially-devised, original evaluation scheme for assessing the effectiveness of any CF-based design adhering recommending registry in any deployment scenario.

The need for such an evaluation scheme was highlighted at the beginning of the chapter,

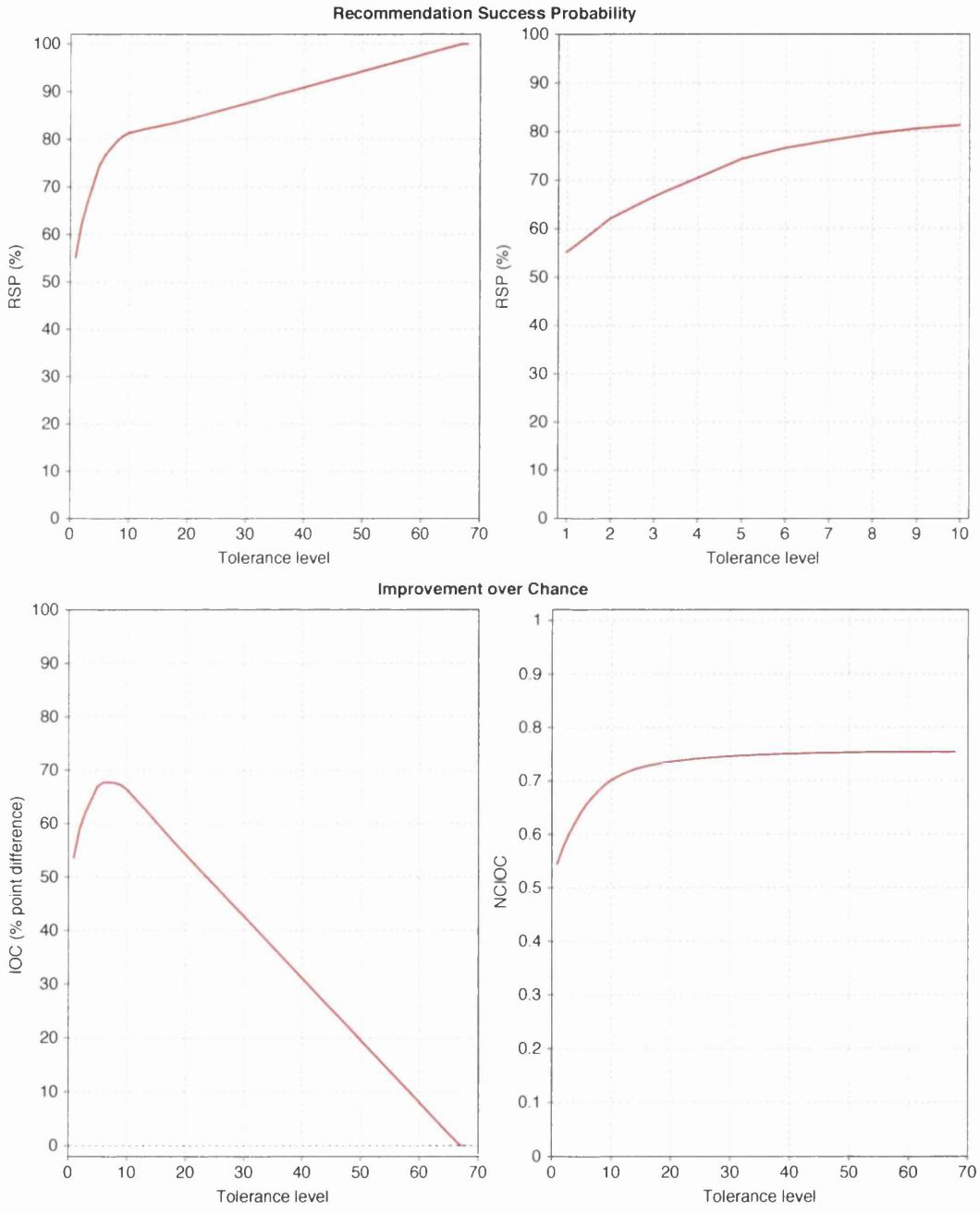


Figure 7.6: An Example Evaluation Presentation

and existing schemes used to evaluate systems that generate forms of service recommendation were subsequently discussed, and discounted in terms of this requirement. My identification of the conceptual similarity between a proposed recommending registry and an IR system, including the notion of recommendation effectiveness, was then noted and my realisation of the relevance of Cooper's IR evaluation explained and justified.

The main body of the chapter has been concerned with the definition and discussion of the various aspects of my evaluation scheme. The concepts of Experienced Service Selection Entries (ESSEs) and tolerance levels have been explained, and the devised effectiveness measures of RSP, IOC and NCIOC have been introduced and defined, both conceptually and mathematically.

Finally, the manner in which the evaluation scheme is used in this research has been explained, and the way in which evaluation results are presented has been specified.

My evaluation scheme is first used in this research in Chapter 9 to evaluate a prototype recommending registry which adheres to my basic CF-based design. The basic design itself is defined in the next chapter.

Chapter 8

A Basic Recommending Registry Design

A basic CF-based recommending registry design is formulated in this chapter, through the development of basic generally-applicable solutions to the previously defined research issues.

8.1 Justifying the Development of an Advanced Registry Design

In order to determine whether my proposed CF-based approach to personalised service recommendation, detailed in Chapter 6, justified development into an advanced general design for a recommending registry, a basic registry design was first developed, and its viability and validity assessed. Only if the basic design did prove viable and valid would further development be justified.

The initial registry design was formulated from my proposed approach through the development and adoption of basic, generally-applicable solutions to the research issues defined in Section 6.2.1. Investigation of the design was then made through the construction of a prototype design-adhering recommending registry for a real-world scenario, which was then evaluated for effectiveness using my ESSE-based scheme defined in Chapter 7.

More precisely, the prototype registry was constructed for the Department of Computing Science at the University of Glasgow, to recommend printers; this investigative scenario will be referred to as the "DCS printer scenario". The basic registry design is

presented in this chapter, and the construction and evaluation of the prototype registry discussed in the next chapter.

8.2 Basic Solutions to the Considered Research Issues

The devised basic, generally-applicable solutions to the first six research issues specified in Section 6.2.1 are defined below; the seventh research issue of evaluating recommendation effectiveness was addressed in Chapter 7. As was stated and justified in Section 6.2.2, less attention was given to the first two research issues, which are concerned with how a service selection history is actually recorded and a requesting consumer's situation attributes acquired. From a research perspective, the other research issues, which are concerned exclusively with the core idea of generating personalised service recommendations from a service selection history, were of greater importance. The solutions relate to how a developer would construct, configure and deploy a recommending registry in a particular service-oriented scenario to recommend particular types of service.

8.2.1 Recording Service Selections

How are service selections and their associated situation attributes recorded? In other words, how is a service selection history recorded? Given the problems associated with manual specification, as discussed in Section 6.2.1, an unobtrusive automated approach in which details are recorded automatically would seem preferable. In many scenarios, service usage activity might already be captured automatically, for reasons of accountability (who was responsible for this particular service use?) or charging (who should be charged for this particular service use?). The core recorded details of a detected service use would probably consist of the date-time of occurrence, the service's ID and the user's ID. A service use obviously corresponds to a prior service selection. Thus, a registry developer could construct a mechanism that automatically captures details of service selections by linking into an existing service usage recording scheme. For every service use detected, a corresponding entry in the service selection history could be recorded, which combined the basic captured details of the service use/selection with situation attributes obtained from other sources of contextual information. This style of approach was adopted in the DCS printer scenario, with the service selection history recorded via a mechanism that linked into an existing print-quota system. More details will be given in the next chapter.

Although I would advocate an automated solution to the recording of a service selection history, a generally-applicable solution cannot be defined. Given that history recording would involve the capture of service selections made in a particular scenario, and the acquisition of particular contextual information, the solution devised by a registry developer must, by necessity, be scenario-specific.

However, it *is* possible to define a generally-applicable format for a recorded service selection history entry. Such a standardised history format enables corresponding generally-applicable solutions to the other research issues to be devised and developed into a general design. Regardless of how a service selection history is recorded, adherence to the standardised history format would allow a developer to construct and deploy a recommending registry according to the remainder of the general design.

I have therefore defined a standardised generally-applicable format for a service selection history. A history entry must record the following named details about a service selection:

- ServiceID - the unique identifier of the selected service.
- UserID - the unique identifier of the individual responsible.
- WhenOccurred - the date-time at which the service was selected.
- ServiceType - the type of the selected service.

These named details will be referred to as "core attributes". The entry must also contain those type-specific situation attributes which record relevant aspects of the situation in which the service selection was made. An entry therefore consists of a set of core and situation attributes. Each attribute has a name that identifies it (e.g. "ServiceType", or "PhysicalLocation"), and a value (e.g ServiceType = Printer, or PhysicalLocation = RoomA). The value of a particular attribute could be one of many (e.g. ServiceType = {Printer, Projector, LightingControl, ...}, or PhysicalLocation = {RoomA, RoomB, RoomC, ...}).

In order to be generally applicable, the value of every attribute, apart from WhenOccurred, will be represented in and interpreted by a recommending registry only as a symbol. This means that the only operation that can be performed on the attribute value is an equality test: is this value the same as that value? No other relationships between attribute values can be determined (e.g. RoomA and RoomB may be adjacent to one another, but

this fact will be undetectable). Symbolic representation and interpretation ensures that any situation attributes chosen to record relevant contextual aspects of the service selection situation can be treated in the same homogeneous way. For example, regardless of whether the situation attributes were the printer-specific attributes of physical location, role and time, in the running illustration of Alice, or the DNA sequence analyser-specific attribute of network location, in the running illustration of Bob, the different attribute values would all be represented and interpreted as symbols.

To clarify, the service selection detailed in Figure 7.1 would be represented as an entry in the service selection history of the grid recommending registry used by Bob as:

```
ServiceID = "B"
UserID = "A29"
WhenOccurred = 16/06/2004 14:32:05
ServiceType = "Seq_Analyser"
NetworkLocation = "130.209.246"
```

The core and situation attribute values of "B", "A29", "Seq_Analyser" and "130.209.246" are all represented as strings, as the string equality operation would enable the values to be treated as symbols, whilst "16/06/2004 14:32:05" is represented as a date-time. The service selection would be interpreted as being made in the particular network location represented by "130.209.246" (an IP subnet). If, however, the attribute value was recorded as "130.209.245", the service selection would be interpreted as being made in another, unrelated network location. The two IP subnets might be very close to one another in network terms, but in symbolic terms, they are completely different (the two strings differ).

8.2.2 Acquiring a Requesting Consumer's Situation Attributes

How are the situation attributes that define a requesting consumer's situation acquired? As with the recording of service selection history, an unobtrusive automated scheme for obtaining such attributes would seem preferable, but could prove complex for a registry developer to construct. A much simpler solution would be to require the requesting consumer to specify his situation attributes manually. Given that the consumer has already explicitly submitted a service request to the recommending registry, he might be willing to specify this additional information as well. This style of approach was adopted in the DCS printer scenario.

However, regardless of whether the attribute acquisition scheme devised by a developer is automated or not, it must, by necessity, be scenario-specific; consumer situation attributes will be particular to the specific deployment scenario and service type required. Consequently, I have addressed the issue of situation attribute acquisition in a similar way to that of service selection history recording, by defining a standardised generally-applicable format for acquired attributes. Once again, regardless of how consumer situation attributes are obtained, adherence to the standardised attributes format would allow a developer to construct and deploy a recommending registry according to the remainder of the general design.

Essentially, the situation attributes of a requesting consumer acquired by the recommending registry must be the same type-specific set as those recorded in any service selection history entry that refers to a service of the requested type, and must adhere to the same standardised format. Thus, each consumer situation attribute must have the same name as the corresponding history entry situation attribute, and its value must be drawn from the same underlying set of values. To be generally applicable, every consumer situation attribute will also be represented and interpreted only as a symbol.

For example, when Bob submits a service request for a DNA sequence-analyser service (type Seq_Analyser) to the grid recommending registry in Figure 7.1, the acquired situation attribute could be:

```
NetworkLocation = "130.209.240"
```

Note that this consumer attribute is the analyser-specific attribute of network location. Moreover, it has the same name as the equivalent situation attribute in the last example of a Seq_Analyser service selection history entry in Section 8.2.1. The network location value is also represented as an IP subnet, in the form of a string. Thus, since they share the same standardised format, in terms of name, value, and symbolic representation and interpretation, the situation attribute of Bob and that of a Seq_Analyser history entry can be easily compared by the grid recommending registry.

8.2.3 Choosing Situation Attributes

How is a decision made as to which set of situation attributes are recorded with the core attributes in a service selection history entry? The choice of situation attributes for each service type is very important, as a recommending registry will identify those relevant

service selections that were made in a situation similar to that of the requesting consumer by comparing the entry-recorded situation attributes of each recent type-matching service selection against the consumer's situation attributes. The service recommendation will be generated through assessment of these relevant service selections.

It was implied in Section 6.1.1 that the situation-similarity of a service selection would be assessed by comparing *all* of its entry-recorded situation attributes against the requesting consumer's situation attributes. For this to happen, the choice of situation attributes would need to be made by the developer when the history recording mechanism is first constructed. For each service type, the developer would need to decide which aspects of a situation would be of most relevance when service appropriateness was being assessed (i.e. those important aspects which individuals would take into consideration when deciding upon, and then selecting, appropriate services of this type). These particular aspects would then be recorded as situation attributes in the history entry of a detected service selection. The history recording mechanism would need to be constructed to acquire this particular contextual information. Although possible, this solution would not be very flexible, and would reduce the potential for variation or experimentation with regard to the assessment of service selection situation-similarity.

A more flexible solution would be for the recommending registry to assess the situation-similarity of a service selection by comparing a *subset* of the entry-recorded situation attributes against the requesting consumer's corresponding situation attributes. This is the solution adopted in this research. When deciding upon situation attributes to record for each service type, the developer would identify *any* situation aspects that might be of some relevance when service appropriateness was being assessed, not just those that appeared to be most relevant. The history recording mechanism would then be constructed to acquire all of this contextual information, and to record the relevant type-specific attributes in the history entry of a detected service selection. The combination of numerous entry-recorded situation attributes and a recommending registry that could assess service selection situation-similarity in terms of a subset of these attributes would provide considerable flexibility. For a particular service type, the developer would be able to configure the registry to generate recommendations using any one of the situation attribute subsets.

As a clarifying example, imagine that the developer of the grid recommending registry used by Bob has decided to adopt this more flexible solution. For the DNA sequence-analyser service type, he has decided to record in an analyser selection history entry the

4 situation attributes of network location (NetworkLocation), the genome type analysed (GenomeTypeAnalysed), the day of the week (DayOfWeek), and the hour of the day (HourOfDay). Although network location is generally considered the most important situation attribute in assessing analyser appropriateness, the developer suspects that the type of genome being analysed might also play a part. Moreover, since network performance may vary on a daily or hourly basis, the developer also records these attributes. Thus, the example of the Seq_Analyser history entry given in the earlier discussion of history recording (Section 8.2.1) now takes the following form:

```
ServiceID = "B"
UserID = "A29"
WhenOccurred = 16/06/2004 14:32:05
ServiceType = "Seq_Analyser"
NetworkLocation = "130.209.246"
GenomeTypeAnalysed = "Mammal"
DayOfWeek = "Wednesday"
HourOfDay = "14"
```

Given that there are 16 possible subsets of the 4 situation attributes (e.g. NetworkLocation, NetworkLocation and GenomeTypeAnalysed, DayOfWeek and HourOfDay, ...), and service selection situation-similarity could be assessed in terms of any of these different subsets, the developer now has significant flexibility when configuring the registry. Through experimentation, the situation attribute subset which would generate the most effective Seq_Analyser recommendations should be found.

8.2.4 Choosing a Length of Time-Window

How is a decision made as to the length of time-window used by a registry to identify the recent relevant service selections? As was explained when discussing this issue in Section 6.2.1, time-windows are used, one for each service type, to enable a recommending registry to respond quickly to any changes that affect service appropriateness. People should factor relevant change into their service selections. Thus, by basing a recommendation only on situation-similar type-matching service selections that occurred recently, within the type-specific time-window, a registry should be able to reflect such change. Theoretically, the shorter the time-window, the more rapidly change should be reflected in generated

recommendations. However, the shorter the time-window, the smaller the number of recent situation-similar type-matching service selections identified, perhaps reducing the effectiveness of generated recommendations generally. The ideal "trade-off" length of time-window is one which would enable the recommending registry to generate the most effective recommendations whilst reflecting relevant change as rapidly as possible.

The most basic solution to this issue of time-window choice is for the developer himself to decide upon the time-window to be used for each service type. This is the solution I adopted in formulating the basic registry design. For a particular service type, the developer must consider how often change that will affect the appropriateness of type-matching services is likely to occur. Such change might include the introduction of new services, or changes in the characteristics of existing services.

If relevant change is a frequent occurrence, the developer might favour a smaller time-window; relevant change would be reflected more rapidly, at the risk of reducing the general effectiveness of generated recommendations. For example, if the grid of services accessed by Bob was relatively volatile, with new services being introduced and the prices of existing services being changed on a weekly basis, the developer of a grid recommending registry might choose a time-window of, say, 2 weeks for the DNA sequence-analyser service type.

If relevant change happens infrequently, the developer might favour a larger time-window instead; the general effectiveness of generated recommendations would be maximised, though relevant change might not be reflected rapidly. For example, in the running example of Alice, change that affected the appropriateness of departmental printers would probably occur infrequently, with new printers being introduced on a monthly to yearly basis, and existing printers becoming faulty (and quickly being repaired) on a monthly basis. The developer of the departmental recommending registry might therefore choose a time-window of, say, 8 weeks for the printer service type.

8.2.5 Identifying Situation-Similar Service Selections

How should situation-similarity be defined in order to identify relevant service selections? Given the symbolic representation and interpretation imposed by the generally-applicable formats of a service selection history entry and a requesting consumer's situation attributes, only one basic definition of service selection situation-similarity is possible.

As was detailed earlier, in generating a recommendation for the service type requested, the registry would assess the situation-similarity of each recent type-matching service selec-

tion by comparing a subset of its entry-recorded situation attributes against the requesting consumer's corresponding situation attributes. The developer would have previously defined the situation attribute subset to be used, for that particular service type. In terms of comparing the two symbol-interpreted values of a particular situation attribute, the only operation that can be applied is an equality test. Therefore, a type-matching service selection is defined as being situation-similar if, for all subset situation attributes, the entry-recorded attribute value equals the corresponding requesting consumer's attribute value. In less formal terms, if the considered subset of situation attributes match exactly, the service selection is considered to have been made in a situation similar to that of the requesting consumer. Thus, such a definition of situation-similarity will identify those service selections that were made in an *identical* situation to the consumer.

Consider how the grid recommending registry would respond to Bob's earlier service request for a DNA sequence analyser (type Seq_Analyser), with one of Bob's situation attributes being:

```
NetworkLocation = "130.209.240"
```

The developer has configured the situation attribute subset used to assess Seq_Analyser service selection situation-similarity to be the NetworkLocation attribute alone. Thus, the relevant recent Seq_Analyser service selections identified will be those that have the same NetworkLocation value of "130.209.240".

8.2.6 Generating a Recommendation

How is a recommendation generated? As was explained previously, the registry responds to a consumer's service request by identifying those service selections that refer to a service of the requested type, and that were recently made in a situation similar to that of the consumer. In more technical terms, the registry identifies those history-recorded type-matching service selections which occurred within the type-specific time-window, and whose situation attributes exactly match those of the requesting consumer, for the type-specific situation attribute subset. These identified recent situation-similar type-matching service selections are called the "relevant" service selections. Thus, rephrasing the original question, how does the registry rank the available type-matching services in an ordered list by collective perceived appropriateness, through assessment of the relevant service selections identified?

The history-recorded form of the relevant service selections encapsulates considerable information about people's recent service selection behaviour. As was detailed earlier, the core attributes of a service selection history entry record the service selected (ServiceID), the individual responsible (UserID), and the date-time of occurrence (WhenOccurred). Viewed as a whole, therefore, the set of relevant service selections show a number of different individuals (each with a different UserID) selecting and using a variety of different type-matching services (each with a different ServiceID) over time (the sequence in which service selections occurred could be obtained by ordering them by WhenOccurred).

Clearly, various recommendation generation algorithms could be devised which each assess this information in a particular way, in order to determine the perceived appropriateness of every available type-matching service and thus rank them in an appropriateness-ordered list. For example, an algorithm might calculate the perceived appropriateness of a service in terms of the number of individuals who had selected it, the recency of its selections, or a combination of factors. However, to formulate the basic registry design, I decided to calculate perceived service appropriateness in terms of two very simple factors: whether a service was selected at all, and, if so, how often. One recommendation generation algorithm was devised that calculated service appropriateness in terms of the first factor, whilst a second was devised that considered both.

The first algorithm, referred to as STR (Selection Tied Ranking), is exceedingly simple in its assessment of the relevant service selections. It is assumed that a selected service is perceived as being more appropriate than one that was not selected. Thus, a recommendation is generated that consists of two tied ranks: a top rank that contains available type-matching services that were selected at least once, and a bottom rank containing the remaining available type-matching services that were not selected at all.

The second algorithm, referred to as SCO (Selection Count Ordering), is a modification of STR. Once again, it is assumed that a selected service is perceived as being more appropriate than one that was not selected. However, it is also assumed that the *number* of times a service was selected provides a direct indication of its perceived appropriateness, in that more selections of a service imply more "votes of confidence". A recommendation is therefore generated by ranking the available type-matching services by the number of times each one was selected, from most to least. As with STR, the bottom rank contains all those services that were not selected at all.

As an example, imagine that Bob has submitted his request for a Seq_Analyser to

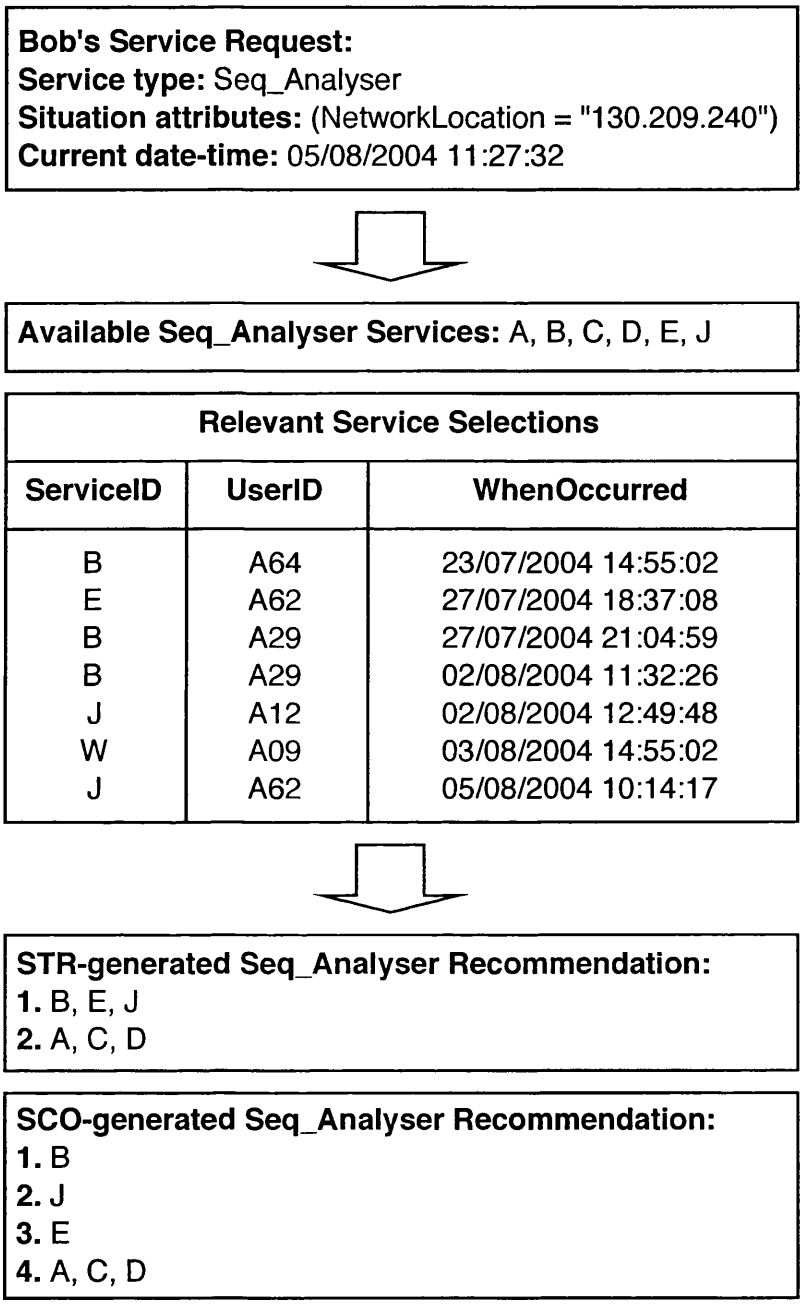


Figure 8.1: The STR and SCO Recommendation Generation Algorithms

the grid recommending registry at 05/08/2004 11:27:32, as illustrated in Figure 8.1. The registry responds by ascertaining that Seq_Analyser services A, B, C, D, E and J are currently available. It also identifies the set of relevant service selections: those selections that refer to a Seq_Analyser service, that had been made in the last 2 weeks (the Seq_Analyser-specific length of time-window), and whose NetworkLocation value matches Bob's NetworkLocation value of "130.209.240" (NetworkLocation is the Seq_Analyser-specific situation attribute subset used to assess situation-similarity). The relevant service selections are shown in the middle of Figure 8.1. Note that B has been selected three times (by users A64 and A29), J twice (by users A12 and A62), and E and W once (by users A62 and A09 respectively). The registry then assesses the relevant service selections, generating a Seq_Analyser recommendation that is returned to Bob. As is shown at the bottom of Figure 8.1, if the registry uses the STR algorithm, B, E, and J would be ranked in first place, and A, C and D in second place; B, E, and J were selected, whilst the other three available Seq_Analyser services were not. If the registry uses the SCO algorithm, B would be ranked first (with 3 selections), J second (with 2 selections), E third (with 1 selection), and A, C and D last (as before). Note that the selected service W is not present in a recommendation, as it is not currently available.

How does the registry rank the available type-matching services if *no* relevant service selections are identified? The basic solution adopted in this research is for a recommendation to be generated that consists of all the services in a single tied rank. This is essentially equivalent to the normal behaviour of a consumer-driven SDM registry, in which available matching services are returned as an unordered set.

8.3 The Basic Registry Design and Developer Tasks

8.3.1 The Basic Registry Design

The basic SDM recommending registry design developed from the foregoing research solutions is defined below. The design is phrased on the operational level, and is based on the abstract model of a recommending registry stated in Section 4.4. It does not specify architectural details of a registry, such as its internal composition of interacting code components, or particular types of data structure used. At this early stage in the investigative process, it is not possible to define such precise details. Rather, the design specifies *how* a registry should operate in generating a personalised service recommendation. A working

recommending registry can be constructed that adheres to this operational design, as is demonstrated by my prototype (Chapter 9).

For the particular scenario in which it will be deployed, a constructed recommending registry must:

1. Have access to a service selection history for the scenario, which adheres to the standardised format defined in Section 8.2.1. The history records details of all service selections that occur within the scenario on a continual basis.

2. Be able to acquire the situation attributes of a requesting consumer. The acquired attributes must adhere to the standardised format defined in Section 8.2.2.

In generating a personalised service recommendation for a requesting consumer, a constructed recommending registry must operate as follows:

1. As defined in the abstract model, the requesting consumer submits a service request to the registry, stating the required service type and any specific attributes. The registry responds by acquiring the type-specific situation attributes of the requesting consumer. It also identifies those available services with the required type by assessing the advertised service descriptions.

2. The registry identifies those relevant service selections that refer to a service of the required type, and that were recently made in a situation similar to that of the consumer. More precisely, three tests are applied to each history-recorded service selection:

    - Type-matching - The ServiceType attribute of the service selection is tested for equality against the required service type specified in the consumer's service request. If the compared values are equal, the service selection is *type-matching.*
    - Situation-similarity - For the type-specific situation attribute subset, each service selection situation attribute is tested for equality against the requesting consumer's corresponding situation attribute. If every pair of compared values is equal, the service selection is *situation-similar.*
    - Recency - The WhenOccurred attribute of the service selection is tested, to determine whether the selection was made within the type-specific time-window. The date-time start of the time-window, TimeWindowStart, can be calculated

by subtracting the time-window length from the current date-time. If WhenOccurred >= TimeWindowStart, then the service selection is *recent*.

The set of relevant service selections identified are those that are type-matching, situation-similar and recent.

3. The registry assesses the relevant service selections to rank the available type-matching services in an ordered list by collective perceived appropriateness. This list is the personalised service recommendation. Either the STR or SCO recommendation generation algorithm, as defined in Section 8.2.6, is used. If no relevant service selections were identified, a recommendation is generated that consists of all the available type-matching services in a single tied rank.

4. The registry returns the generated personalised service recommendation to the requesting consumer. As required by the abstract model, the recommendation is filterable to show only those services that also matched the consumer-specified service attributes.

8.3.2 The Tasks of the Registry Developer

For a design-adhering recommending registry to become operational in a particular scenario, the developer must therefore perform the following sequence of tasks:

1. Service selection history recording - The developer must construct the service selection history recording mechanism. For each type of scenario service, he must decide which situation attributes to record in a service selection history entry along with the core attributes. As was advocated in Section 8.2.1, the history recording mechanism should ideally be automated.

2. Recommending registry construction - The developer must construct a recommending registry that adheres to the design specified above.

3. Recommending registry configuration - The developer must configure the constructed registry. For each type of scenario service, he must decide on the subset of type-specific situation attributes to be used in assessing service selection situation-similarity, and on the length of time-window to be used in assessing service selection recency (as discussed in Sections 8.2.3 and 8.2.4 respectively).

4. Recommending registry deployment - The developer must deploy the registry. This can occur once the history recording mechanism has recorded a sizeable service selection history, which the registry can use in generating personalised service recommendations. Through the registry's lifetime, the developer may wish to reconfigure it periodically, so as to "tune" the effectiveness of its recommendations.

8.4 Summary

In this chapter, details have been given of the basic, generally-applicable solutions that were developed to address the various research issues specified in Section 6.2.1. A basic design for a CF-based recommending registry has then been detailed based on the research solutions, and the tasks that a registry developer needs to perform for a registry to become operational have been given.

This basic design was used to investigate whether the proposed CF-based approach should be further developed into an advanced design. This was done through the construction and evaluation of a prototype recommending registry that adheres to the basic design, to determine design viability and validity. The next chapter is concerned with this aspect of my research.

Chapter 9

Assessing the Basic Recommending Registry Design

In this chapter, details are given of the assessment of the basic CF-based recommending registry design, in terms of viability and validity, through the construction and evaluation of a design-adhering working prototype registry in a real-world scenario.

9.1 Background to Assessment

Investigation into the viability and validity of the basic recommending registry design, detailed in Chapter 8, was made through the construction, and subsequent evaluation, of a design-adhering working prototype registry. This prototype was constructed to recommend printers in the Department of Computing Science at the University of Glasgow (the DCS printer scenario). The basic design could be deemed viable if it enabled the prototype to generate personalised service recommendations from a recorded service selection history. It could be deemed valid if these generated recommendations were actually effective. If the basic recommending registry design was demonstrated to be both viable and valid, its development into a more advanced version would be worthwhile.

Details of the DCS printer scenario, the constructed prototype registry and its associated service selection history, and the setup of the effectiveness evaluation are given below.

9.1.1 The DCS Printer Scenario

The Department of Computing Science occupies (partly or wholly) three buildings on the Glasgow University campus, and consists of academics, researchers, teaching and tutoring staff, administrative and systems support staff, postgraduate students (PhD and MSc) and undergraduate students. In a typical month, the department also contains a small number of transient visiting speakers and researchers, academics on sabbatical, and exchange students. To provide a rough indication of size, 574 different individuals were recorded as using departmental printers in January 2004, and 778 individuals in February 2004.

The department therefore typifies a small to medium-sized organisation with a constant stream of uninformed consumers who need to use services that are available on the organisational computing network. As with any organisation, long-term members leave and uninformed replacements arrive (e.g. newly appointed academics and students starting courses), whilst uninformed visitors come and go. The department could well be the one which Alice was visiting! The organisational nature of the department also means that a significant proportion of service selections made in the scenario should be appropriate choices; long-term departmental members should be able to make such appropriate selections, given their experience.

From the perspective of assessing a design-adhering recommending registry, one type of service is as representative as any other, given that recommendations for all types are generated in exactly the same manner. This research focuses on printers, which are currently one of the few types of service in universal use. Many organisations make use of numerous shared printers that are available on an organisational network, as this is more cost-effective than supplying every staff member with a printer. This arrangement also makes print-quotas possible. Moreover, printers are one of the main service types cited in the service discovery and ubicomp research literature [1, 7, 19, 27, 38, 44, 50, 53, 56, 59, 66, 72, 78, 94, 95, 123].

There are approximately 80 printers in the DCS printer scenario, distributed across all three buildings. Some printers are in private offices, and are considered of restricted access. Others are designated for general use, and are positioned in corridors. Some printers are in general printing/photocopying rooms which are locked outside normal working hours. Yet others are designated primarily for student use, and are located in student laboratories. Some printers are faster than others, whilst some printers break down on a relatively frequent basis. In such a scenario, an uninformed consumer could face consid-

erable difficulty in finding and selecting an appropriate printer to use. The DCS printer scenario was therefore a suitable environment in which to investigate a design-adhering recommending registry.

9.1.2 The Constructed Working Prototype Registry

Recording the Service Selection History

As defined in the sequence of developer tasks specified in Section 8.3.2, I first constructed the mechanism to record the service selection history. For the registry to be able to recommend printers, the mechanism needed to detect and record every printer selection made within the department. Fortunately, at the time when the history-recording mechanism was being constructed, the departmental printing architecture was being modified by systems support staff to enable a print-quota system to be deployed. The architecture was rearranged so that any print job (i.e. a document to be printed) sent by an individual to a network-accessible printer was channelled through one of four print-servers. Three of these print-servers were subsidiary, forwarding received jobs onto the other, main, print-server, which in turn sent each job to the specified printer. This arrangement is illustrated in Figure 9.1. The support staff deployed a print-quota system, Print Manager Plus [62], on the main print-server, which logged details of every job received, processed and forwarded. For a particular print job, these details included the departmental user-name of the individual responsible (the "login" of his account in the departmental computing environment), the name of the printer to which the job was sent, the date-time of occurrence, the host-name of the departmental computer used by the individual (if detectable), and the byte size and page length (if detectable) of the document involved. It was therefore possible to construct an unobtrusive automated history-recording mechanism which made use of this print job log.

A submitted print job corresponds to a printer selection. Thus, the constructed mechanism periodically parsed the log of the quota system and, for each newly-logged job, recorded a corresponding entry in the service selection history. This process enabled the core attributes of a service selection to be recorded; ServiceID was set to the logged printer name, UserID to the logged departmental user-name, WhenOccurred to the logged date-time of occurrence, and ServiceType obviously to "Printer". Although the job log of the quota system on the main server recorded a significant proportion of departmental printing activity, it did not record all of it. For legacy reasons, the three subsidiary print-servers

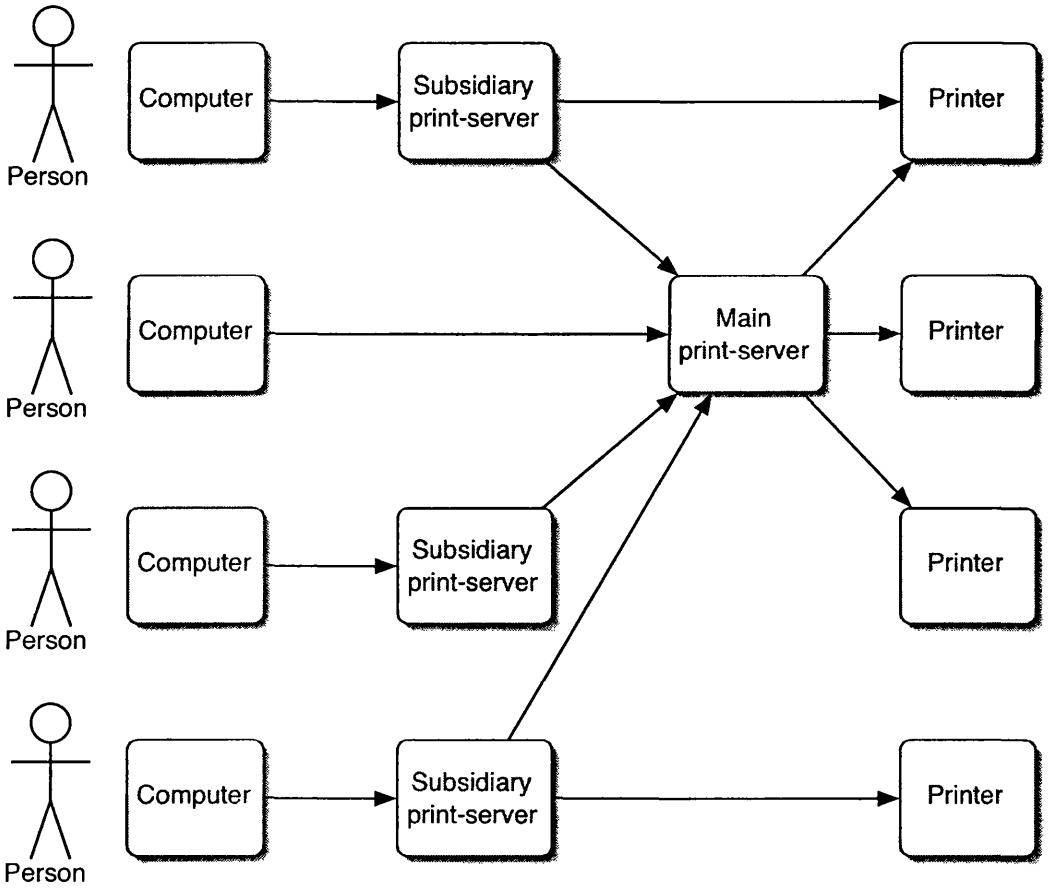


Figure 9.1: The Departmental Printing Infrastructure

also sent some jobs to certain printers directly, rather than forwarding them to the main printer-server. Moreover, it was discovered that the quota system did not always log those jobs that *were* forwarded. However, further investigation uncovered the fact that the three subsidiary print-servers (two Windows and one Unix CUPS [97]) did log those print jobs which they themselves received, processed and forwarded, recording roughly the same details as the quota system log on the main print-server. Thus, to record as many departmental printer selections as possible, the history-recording mechanism was actually constructed to collate (e.g. remove duplicate entries) and parse all four print job logs from the four different print-servers periodically.

With regard to the situation attributes recorded along with the log-obtained core attributes of a printer selection, the history mechanism was initially constructed to acquire a variety of situation aspects which appeared of some direct or indirect relevance when printer appropriateness was being assessed: the title (dr,mr,mrs, ...), departmental role (academic, support staff, ..), research group (if any) and physical location (room A, room B, ..) of the selecting individual, the operating system (windows, unix, ..) and host-name of the computer used by him, the hour of day and day of the week when the selection was made, and the size, type (txt, jpg, pdf, ..) and page length of the document involved.

However, after further assessment, the decision was made to focus only on the three situation attributes which seemed most relevant in this scenario: the selecting individual's physical location (named "Location") and departmental role (named "Role"), and the hour of day (named "HourOfDay"). Given that the departmental printers are distributed over three buildings, which each contain multiple floors, sets of stairs, corridors, rooms etc, location would seem to be an important factor in determining printer appropriateness. With regard to role, although theoretically anyone can use any departmental printer, there is an underlying organisational culture which affects printer usage. For example, undergraduates are not encouraged to use printers in an administrative staff office. Thus, departmental role can also be considered an important factor. Finally, hour of day is important because printer load varies throughout the day, and certain printers (being behind locked doors) are inaccessible outside working hours. Considering only three situation attributes made the research more tractable, as there were consequently only 8 possible attribute subsets with which the recommending registry could be configured to assess printer selection situation-similarity.

For every printer selection obtained from the processed print job logs, the 3 situation

attributes were acquired and recorded in the following manner:

- Location - The physical location of the selecting individual was recorded as the name of the departmental room in which he was estimated to have selected the printer; each room has a different assigned name. Given that the locations of people in the department are not explicitly tracked (e.g. using Active Badges [117]), the selecting individual's location was estimated in one of two ways. Firstly, the host-names of certain departmental computers refer to the rooms in which they are placed. For example, computers in room "bo715", an undergraduate student laboratory, have names such as "bo715-5-05", or "bo715-2-06". Thus, if the parsed print job log had recorded the host-name of the computer used by the selecting individual, this was analysed to see if it referred to a room. If it did, the Location situation attribute was set to the derived room value (e.g, "bo715" in the example above). If this approach failed, a second was tried. The UserID of the selecting individual, his departmental user-name, was used to look up a regularly maintained internal departmental database that recorded the roles and allocated offices of permanent departmental members such as academics and PhD postgraduate students. If the individual had a database entry, which recorded his allocated room, the Location situation attribute was set to this value. Given that a permanent departmental member mainly works in his allocated office, it is reasonable to assume that his printer selection was made in that location. If both approaches failed, Location was set to an unknown "null" value. In the time over which the service selection history was acquired, printer selections were recorded as being made in 83 different rooms.

- Role - The departmental role of the selecting individual was recorded as one of 14 values, which are all self-explanatory: "academic", "admin_staff", "it" (MscIT postgraduate student), "post_grad", "researcher", "research_fellow", "systems_staff" (systems support staff), "teaching_staff", "tutoring_staff", "undergrad_level1", "undergrad_level2", "undergrad_level3", "undergrad_level4", and "visitor". As with physical location, role was obtained in one of two ways. Firstly, the same internal departmental database for permanent member details was looked up, using the UserID / departmental user-name of the selecting individual. If the individual had a database entry, which recorded his role, the Role situation attribute was set to this value. If this approach failed, the UserID was used to look up the individual's Unix group. When a login account for an individual is created in the departmental

computing environment, it is assigned to a particular group, which determines access rights. Some of these Unix groups are directly equivalent to the named roles above. For example, first year undergraduate students, referred to as "undergrad_level1" above, are members of the "level1" Unix group. Thus, if the retrieved group of the selecting individual was such a group, the Role situation attribute was set to the corresponding named role (e.g. "undergrad_level1" in the example above). If both approaches failed, Role was set to an unknown "null" value.

- HourOfDay - The hour of day was recorded as one of 24 different values: "0" (00:00:00 to 00:59:59), "1" (01:00:00 to 01:59:59) to "23" (23:00:00 to 23:59:59). It was obtained through a simple analysis of the log-obtained WhenOccurred attribute of the service selection.

As an example, an actual printer selection recorded by the service selection history recording mechanism is given below; to preserve anonymity, the UserID value has been changed:

```
ServiceID = "lwf164"
UserID = "anon"
WhenOccurred = 08/03/2004 16:04:50
ServiceType = "Printer"
Location = "g141"
Role = "postgrad"
HourOfDay = "16"
```

The history-recording mechanism was constructed in mid-2003. From late September 2003 to early April 2004, the mechanism processed the print job logs of the four print-servers every week, adding newly-made printer selections (i.e. those which had occurred since the last log processing) to the growing service selection history. The history was stored in a MySQL database table, with the table containing a column field for each core and situation attribute of a history entry; all fields were of SQL type text (i.e. string), apart from WhenOccurred, which was of SQL type date-time. Thus, each table row contained a service selection history entry. In total, 118856 printer selections were recorded (some of these were made as early as February 2003, and were added on the first log processing in September). These involved 1114 different individuals using 93 different printers. If a

situation is defined as a unique combination of values for Location, Role and HourOfDay, then printer selections were recorded as being made in 1585 unique situations.

The Constructed Registry Itself

Although the CF-based recommending registry design is phrased in terms of its being an augmented form of a consumer-driven SDM registry, the constructed prototype recommending registry itself does not use advertised service descriptions in generating personalised service recommendations, as there was no existing departmental SDM registry to use as a prototype building block. Moreover, as this research is primarily concerned with using service selection history to generate recommendations, advertised service selections are of little importance.

In the posited CF-based recommending registry design, advertised service descriptions are used in two places in generating a personalised service recommendation. At the beginning of the process, the descriptions are used to identify those type-matching services that are currently available. At the end, they are used to make the generated recommendation filterable, to show only those available type-matching services that also match the attributes specified in the consumer's service request. With the prototype, the first function is approximated by assuming that the type-matching services (departmental printers) currently available are those which were recorded in the service selection history as used in the last 12 weeks. This is a reasonable assumption to make, as the DCS printer scenario is a stable environment, with almost constant availability of printers. The second function is not implemented, as the filterability of a generated recommendation is something of an optional feature, and is not considered of primary importance. These two factors aside, the prototype does operate according to the basic registry design specified in Section 8.3.1.

The prototype recommending registry was written in Java. Given that it was to serve as an investigative tool for researching the CF-based approach to personalised service recommendation, the registry was constructed with flexibility in mind. Thus, the main functional elements of the registry were constructed as independent components, and any one component can be replaced without affecting any other aspects of the registry. It was possible to experiment with a particular component, by devising different implementations and then substituting and assessing each in the registry.

Figure 9.2 shows the core structure of the prototype recommending registry. Data components are named in italics, whilst functional components are named in non-italics.

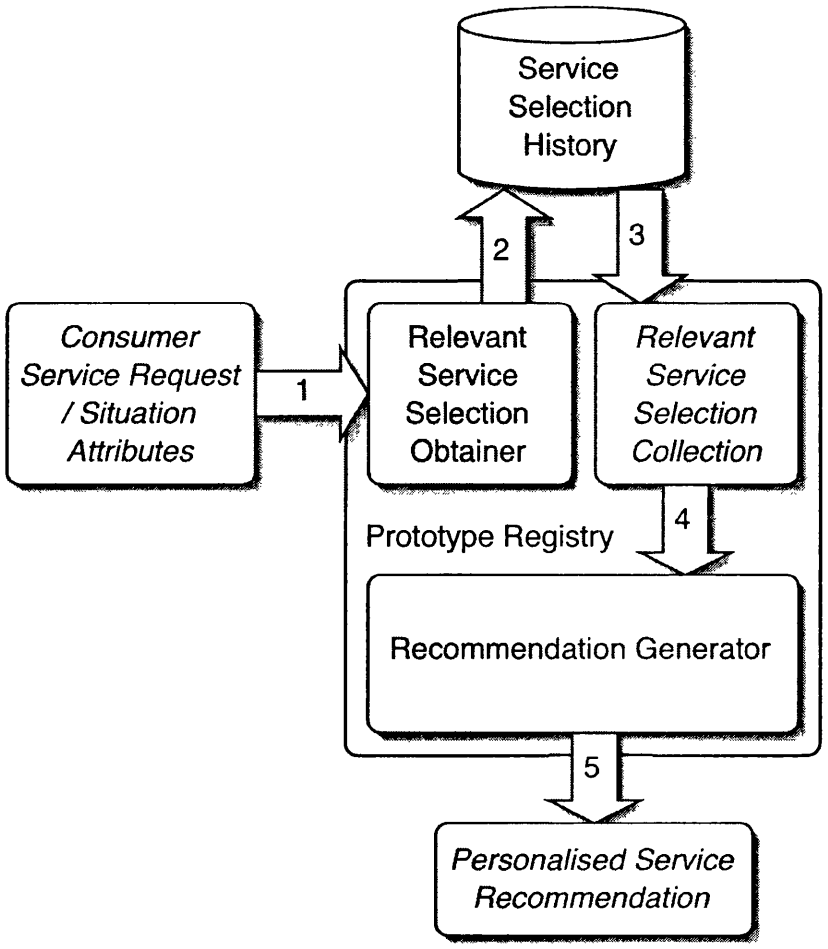


Figure 9.2: The Core Structure of the Prototype Recommending Registry

The numbered arrows indicate the stages of the path taken through the registry in generating a personalised service recommendation. Registry components will be explained in terms of their role in this path:

- Consumer service request / situation attributes submission - A consumer requests a recommendation of departmental printers by submitting a form of service request to the registry (stage 1). The request does not contain a service type or specific attributes; the requested service type is implicit (type Printer), and service attributes are unnecessary, given that the filterability of a generated recommendation is not implemented. The request does contain the printer-specific situation attributes of the consumer: his physical location (Location), departmental role (Role), and the hour of day (HourOfDay). These Consumer Situation Attributes are implemented

as a hash-table, with the name of each attribute mapping to the attribute's value (represented as a String).

- Relevant service selection identification - Inside the registry, the Relevant Service Selection Obtainer assesses the submitted Consumer Situation Attributes, the printer-specific time-window length, and the printer-specific situation attribute subset, and constructs a data-request that will retrieve the relevant service selections from the service selection history, in the form of their ServiceID, UserID and WhenOccurred attributes. Since the history is stored in a MySQL database table, the request is an SQL SELECT statement, with the three tests for service selection relevance (type-matching, situation-similarity and recency) being specified as a conjunction of conditions in the WHERE clause. Each test takes the form defined in the basic registry design.

  For example, imagine that a consumer submitted a printer request, at 10/08/2004 14:06:42, with situation attributes of:

```
HourOfDay = "14"
Location = "f091"
Role = "academic"
```

  Also imagine that the registry was configured with a time-window of 8 weeks and a situation attribute subset of HourOfDay and Role. Then, with the start of the time window being 15/06/2004 14:06:42 (current time minus time-window length), the constructed SELECT statement would be:

```
SELECT ServiceID,UserID,WhenOccurred FROM ServiceSelectionHistory
WHERE (ServiceType = "Printer")
AND ((HourOfDay = "14") AND (Role = "Academic"))
AND (WhenOccurred >= "2004-06-15 14:06:42")
```

  The type-matching test is superfluous in this scenario, given that the service selection history only records printer selections. However, it does demonstrate how type-matching selections would be identified, if selections of multiple service types were recorded.

  The Relevant Service Selection Obtainer submits the constructed data-request to the service selection history database (stage 2). The history database returns to the

registry the requested attributes of the relevant service selections, which are stored in a Relevant Service Selection Collection (stage 3).

- Recommendation generation - The Relevant Service Selection Collection is passed to the Recommendation Generator (stage 4). The Collection stores the service selections in a data-structure optimised for the particular operation of the Generator. The Generator analyses these relevant service selections, generating a Personalised Service Recommendation by ranking the available type-matching services (departmental printers) according to collective perceived appropriateness. As was noted earlier, the available printers are assumed to be those that were recorded in the service selection history as used in the last 12 weeks. Two implementations of the Generator component have been implemented: one that implements the STR algorithm, and another that implements the SCO algorithm. As specified by the basic registry design, if no relevant service selections were identified (the Collection is empty), any Generator component will generate a recommendation that consists of all the available type-matching services in a single tied rank.

- Recommendation output - The Personalised Service Recommendation (of departmental printers) is returned to the consumer (stage 5). The Recommendation is implemented as an ordered array of Service ID arrays. The core array represents the different ranks of the recommendation, with an array element referencing an unordered array containing the ServiceIDs of services with that particular rank.

Although the design-adhering recommending registry prototype was constructed to recommend printers in the DCS scenario, it could have been used to recommend any other type of service, in any other scenario, with minimal modifications. The prototype operates without any understanding of the service selection history's content with which it generates recommendations, apart from the fact that it conforms to the design-specified, symbol-interpreted history format. Thus, from the perspective of the registry, it could have been recommending printers, or projectors, or DNA sequence-analysers, or any other type of service, in any scenario.

Owing to the nature of the investigative DCS printer scenario, with selections of services of only one type (printers) being recorded in the service selection history, the constructed prototype did not enable the consumer to request a particular service type to be recommended: the prototype could only recommend one! However, if selections of

multiple service types were recorded, it would be simple to modify the prototype so that the consumer could specify a particular service type in the submitted service request. This service type could then be used in the type-matching test of the constructed relevant service selection data-request, and the corresponding service recommendation generated.

9.1.3 The Setup of the Evaluation

The prototype recommending registry was evaluated for effectiveness using my ESSE-based scheme defined in Chapter 7. More precisely, the registry was evaluated in terms of the printer recommendations it generated in response to ESSEs (Experienced Service Selection Entries; see Section 7.4.1) identified between 4th January 2004 00:00 and 31st January 2004 00:00. During this time period, 12120 printer selections were made, involving 573 different individuals using 56 different printers. ESSEs were chosen from the month of January 2004 as, by then, departmental printer selections had been recorded for at least 3 months (history recording began in late September 2003). Thus, the generated ESSE recommendations could be based on a reasonably-sized service selection history.

Setting the ESSE Condition

To identify ESSEs, the ESSE condition needed to be defined. For a considered printer selection, this condition is the minimum number (m) of past printer selections (and corresponding uses) that the selecting individual had to make in the last n days for him to be considered experienced enough, and the selection an ESSE. The figure of 28 days was chosen for n. As has already been noted, change is an infrequent occurrence in the DCS printer scenario. Thus, someone who had used departmental printers in the last 28 days should have some relevant prior experience when making a printer selection.

Choosing m was slightly more complex, and was done empirically in this research. The value of m needed to be large enough in order for the set of ESSEs identified to be appropriate printer selections; the more times a selecting individual had recently used department printers, the more experience he should have, and the more likely he would be to make an appropriate choice. However, the value of m also needed to be small enough so that the ESSEs identified were representative of departmental printing behaviour, and thus of the printer requests that uninformed consumers might submit to the recommending registry. Ideally, the ESSE set should refer to as many different individuals (in different situations) and printers as possible.

To explore the consequences of choosing different values of m, 9 ESSE sets were first identified by setting m to values between 0 (least strict) and 40 (most strict), at increments of 5; n was set to 28 days in all cases. These different sets were then assessed in terms of their size, and the number of different individuals and printers they referred to. Figure 9.3 plots, for each ESSE set identified using a different value of m (the x axis), the number of selecting individuals that it refers to (the y axis). Graphs for the number of printers each ESSE set referred to, and the number of selections it contained, are given in Appendix B, in Figures B.1 and B.2 respectively.

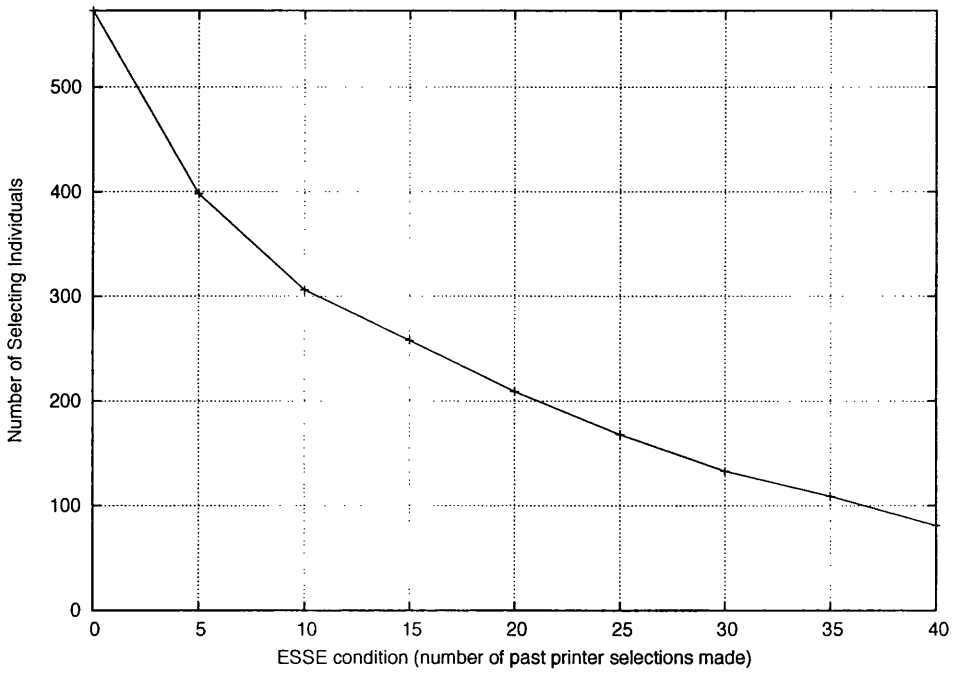


Figure 9.3: The Number of Selecting Individuals Referred to by an ESSE Set

Through consideration of the DCS printer scenario, and assessment of Figures 9.3, B.1 and B.2, it was decided to set $m = 5$. Hence, it was assumed that someone who had selected and used departmental printers at least 5 times in the last 28 days had enough awareness and understanding to make an appropriate printer selection. Even if he had initially selected inappropriate printers, such as one in someone else's private office, or a slow one, by the sixth attempt he should have gained enough experience to make an

appropriate choice.

Figure 9.3 shows that, when $m = 5$, 398 selecting individuals were referred to in the identified set of ESSEs. This is 69.46% of the maximum number possible, as 573 different individuals made printer selections during the considered time segment. In other words, 175 individuals, almost one third of those who made printer selections, were disregarded. Presumably, these people were less likely to have made appropriate printer selections. However, despite ignoring this number of individuals, the $m = 5$ ESSE set would seem to strike the best balance between containing appropriate printer selections and being representative of departmental printing behaviour. At $m = 10$, only 306 selecting individuals, or 53.4% of the maximum, were referred to in the identified set of ESSEs; 267 were not. After $m = 10$, the proportion decreases on an almost linear basis. Thus, for the values of $m >= 10$, the identified ESSE sets cannot be considered representative of departmental printing behaviour, given that the behaviour of so many individuals (potentially all in different situations) is being ignored.

Figures B.1 and B.2 show that the $m = 5$ ESSE set refers to 54 different printers (out of a maximum of 56 used in the considered time segment), and contains 10397 printer selections (out of a maximum of 12120). Consequently, by evaluating the prototype registry using such a representative ESSE set, it should be possible to determine how well it generally performs in the DCS printer scenario.

In summary, the prototype recommending registry was evaluated using the set of ESSEs identified between 4th January 2004 00:00 and 31st January 2004 00:00, with the ESSE condition defined as $m = 5$ and $n = 28$ days. The ESSE set consisted of 10397 printer selections, involving 398 individuals using 54 printers.

Handling Incomplete ESSEs

For a small number of these ESSEs, it had not been possible for the history recording mechanism to record Location and/or Role situation attributes. For 2.32% (241) of these service selections, Location was not recorded, and for 0.36% (37), Role was not recorded. This is simply a consequence of the way in which the Location and Role attributes were acquired by the mechanism. For example, as was detailed in Section 9.1.2, the Location of a printer selection was estimated using two approaches; if neither succeeded, it was not possible to record the attribute. If the prototype registry was configured with a situation attribute subset involving Location or Role, then in generating a recommendation for

such an ESSE, the relevant service selection data-request could not be constructed due to the lack of required situation attributes. Given that no relevant service selections could therefore be identified, it was decided that, in such cases, a recommendation should be generated that consisted of all the available type-matching services (departmental printers) in a single tied rank. This is the same approach as that taken when a data-request *could* be constructed, but no relevant service selections were identified in the service selection history.

The Range of Tolerance Levels

For the 10397 ESSE recommendations generated, the minimum length of recommendation was 66, and the maximum was 68 (i.e. the minimum number of departmental printers ever available was 66, and the maximum was 68). The values of RSP (Recommendation Success Probability; see Section 7.4.3), IOC (Improvement over Chance; see 7.4.4) and NCIOC (Normalised Cumulative IOC; see Section 7.4.5) were therefore calculated for every tolerance level between 1 and 68.

Setting the Tolerance Level for NCIOC Assessment

It was decided to assess the NCIOC value of the prototype registry at tolerance level 5. That is, a consumer would only be willing to consider a maximum of 5 top-ranked printers in a recommendation before giving up. In the DCS printer scenario, many consumers are required to walk to a printer to collect their printed documents after printer selection and use. A consumer would thus be likely to discard a personalised service recommendation quickly if it directed him to various highly ranked printers that proved inappropriate.

9.2 Evaluation of Prototype Registry Effectiveness

As the working prototype recommending registry was the first manifestation of my proposed CF-based approach to personalised service recommendation, initial experiments were of an entirely exploratory nature. It was necessary to ascertain whether the prototype registry could, in fact, generate effective recommendations and, if so, how effective the registry could be, in order to assess the validity of the basic registry design. Hypothetically, the choice of the situation attribute subset used to assess printer selection situation-similarity and the length of time-window used to assess printer selection recency could be

expected to have a definite impact on the effectiveness of the registry-generated printer recommendations. The choice of recommendation generation algorithm, in this case STR or SCO (Selection Tied Ranking or Selection Count Ordering; see Section 8.2.6), could also be expected to affect effectiveness. In order to explore these issues, the prototype registry was configured in a variety of ways, and evaluated in terms of the 10397 ESSE recommendations generated, in three experiments.

Presentation of Evaluation Results

For each experiment, the evaluation results for the various registry configurations assessed are presented in the format of four graphs, as specified in Section 7.5, together with an information table. As an explanatory example, consider Figure 9.4, which presents the results for Experiment One, in which 8 registry configurations were evaluated.

Graphs To recap, the top two graphs plot Recommendation Success Probability (RSP) against tolerance level; the top left plots RSP for tolerance levels 1 to 68, whilst the top right "zoomed-in" graph plots RSP for tolerance levels 1 to 10. The bottom left graph plots Improvement over Chance (IOC), as a percentage point difference, for tolerance levels 1 to 68. The bottom right graph plots Normalised Cumulative Improvement over Chance (NCIOC), also for tolerance levels 1 to 68. The evaluated registry configurations have all been plotted on the graphs, for easy comparison.

Table The bottom table provides more detailed information on the registry configuration evaluations. For a particular configuration, the "Recommendations" columns provide information on the ESSE recommendations generated by the registry. The "Success" column records the number of recommendations that were generated through assessment of a non-zero number of identified relevant service selections. The two "Failure" columns record the number of those that were not. More precisely, the "No service selections" column refers to the number of ESSE recommendations in which no relevant service selections were identified in the service selection history. The "No user situation" refers to the number of those in which no relevant service selections could be identified, as a lack of required situation attributes prevented the construction of the selection data-request. As was noted earlier, in both such cases, the recommendation generated consisted of all the available type-matching services (departmental printers) in a single tied rank.

For a particular configuration, the "Successful Recommendations Info" columns provide information on the identified relevant service selections of the successful ESSE recommendations. The "Mean # Service Selections" column records the average number of relevant service selections on which a successful recommendation was based. The "Mean # Users" and "Mean # Services" record the average number of different selecting individuals and services these relevant service selections referred to.

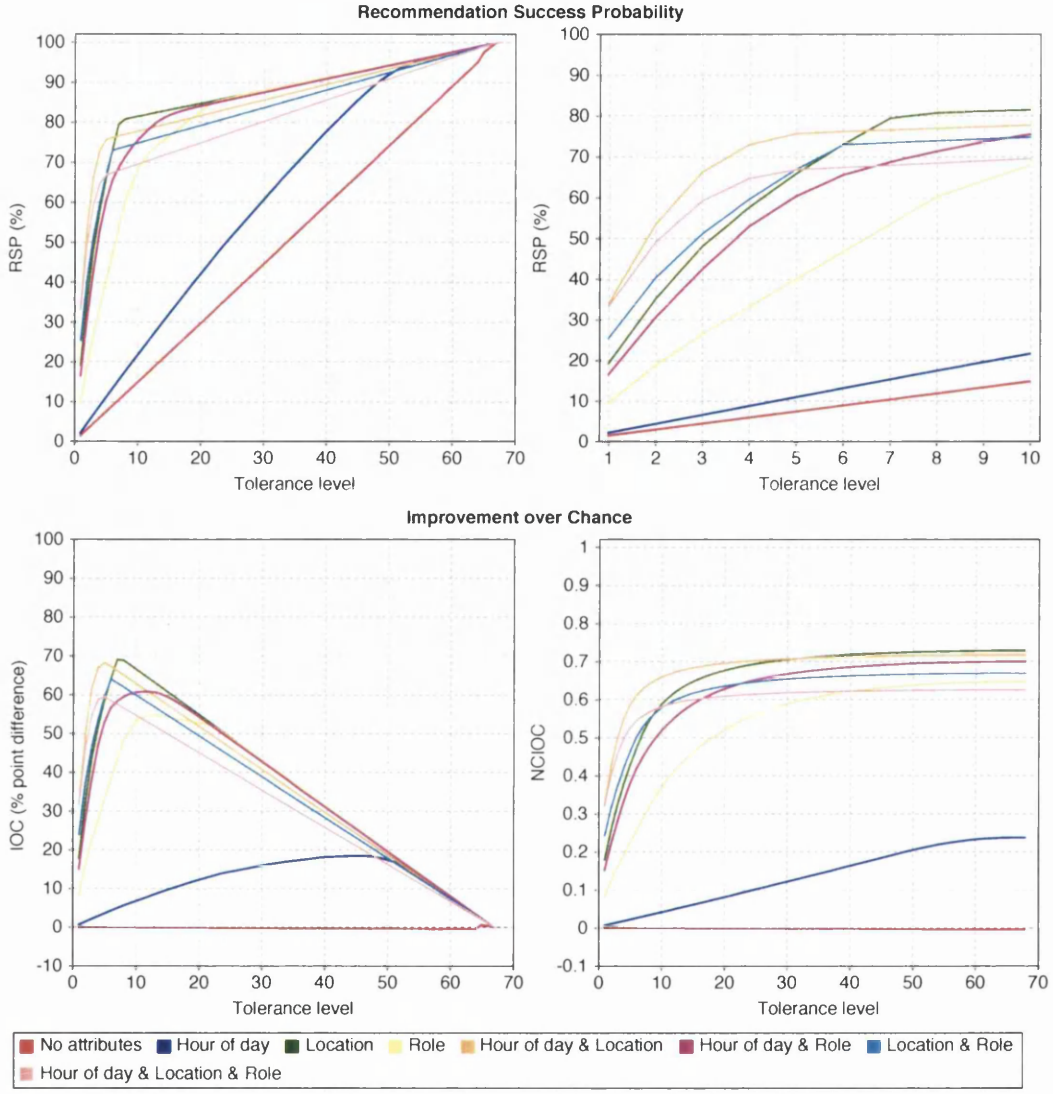
For a particular configuration, the "NCIOC" column records the calculated NCIOC value, at tolerance level 5. To the left of this column, the "Cumul. IOC" column records the equivalent cumulative IOC value (i.e. pre-normalised). By comparing the NCIOC values (or cumulative IOC values) of the different configurations, it is possible to identify which configuration was most effective. The "Rank" column records the rank of the configuration, if the evaluated configurations are ordered by NCIOC value, highest to lowest. The configuration with rank 1 was most effective.

The Importance of NCIOC

Despite the many table-recorded details of the registry configuration evaluations, however, much of the information is not relevant at this stage. The most important issue is the effectiveness of the different registry configurations, which can be assessed using the last three table columns, particularly the "NCIOC" column. Recollect that an NCIOC value greater than 0 indicates that the registry-generated recommendations were more effective than randomly-ordered ones. The higher the value (up to a theoretical "perfect" maximum of 1), the more effective the registry.

9.2.1 Experiment One - Can the Registry Generate Effective Recommendations?

The aim of this initial experiment was to ascertain whether the working prototype registry was able to generate effective recommendations for all or any of the 8 possible situation attribute subsets of HourOfDay, Location and Role. As such, the registry was evaluated for 8 configurations, which differed in terms of situation attribute subset but which all used the same length of time-window and recommendation generation algorithm. Given that change is an infrequent occurrence in the DCS printer scenario, a time-window of 12 weeks was deemed suitable. As this was a pilot experiment, the more basic recommendation generation algorithm of STR was used. The results are shown in Figure 9.4.



Situation Attribute Subset	Recommendations			Successful Recommendations Info			Cumul. IOC (level 5)	NCIOC (level 5)	Rank
	Success	Failure		Mean # Users	Mean # Services Selected	Mean # Service Selections			
		No service selections	No user situation						
No attributes	10397	0	0	868.54	64.96	39342.84	-0.11	-0.0002	8
Hour of day	10397	0	0	460.67	45.83	3257.31	10.53	0.022	7
Location	8737	1419	241	63.56	5.29	3449.91	203.71	0.4265	4
Role	10360	0	37	75	13.96	4865.22	105.62	0.2211	6
Hour of day & Location	8380	1776	241	42.7	2.83	324.84	280.1	0.5864	1
Hour of day & Role	10268	92	37	51.31	8.32	422.77	180.6	0.3781	5
Location & Role	7824	2332	241	64.32	4.1	3522.77	221.28	0.4633	3
Hour of day & Location & Role	7288	2868	241	46.07	2.62	349.51	250.91	0.5253	2

Figure 9.4: Experiment One - Can the Registry Generate Effective Recommendations?

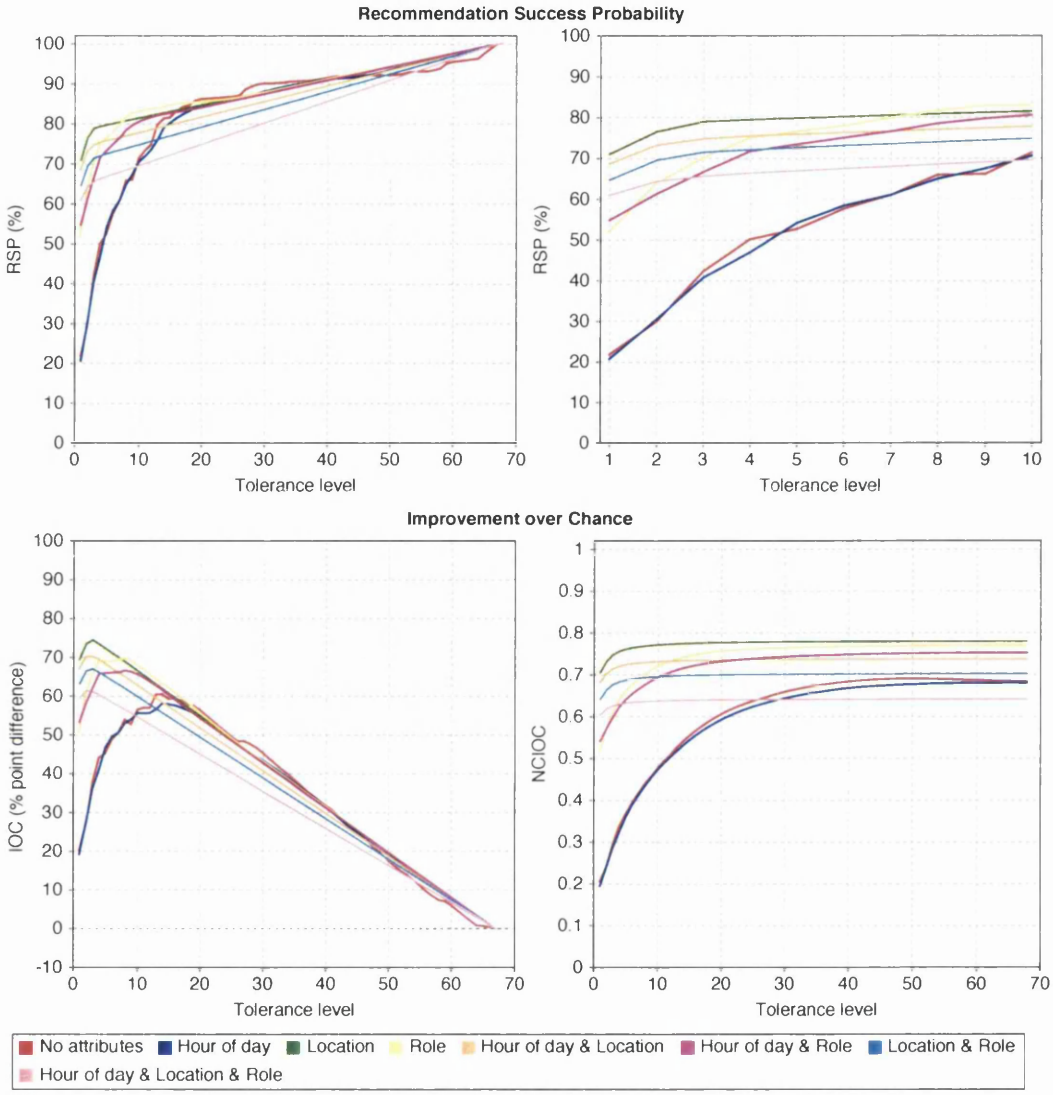
Encouragingly, even at this early stage, 7 of the 8 registry configurations evaluated did generate effective recommendations that were an improvement over chance, having NCIOC values greater than 0. Moreover, the NCIOC values of the 3 highest-ranked configurations are well above 0, ranging from 0.4633 to 0.5864, although some way off a "perfect" 1. In terms of RSP, a consumer with tolerance level 3 had a 66.34% chance of finding an appropriate service with the top-ranked registry configuration (HourOfDay & Location). At tolerance level 5, his chances had increased to 75.72%. Also at tolerance level 5, the RSPs of the second and third-ranked registry configurations (HourOfDay & Location & Role, and Location & Role) were 66.91% and 66.98% respectively.

9.2.2 Experiment Two - What Impact Does SCO Have on Registry Effectiveness?

Following on from the first experiment, the second experiment was set up to ascertain whether the use of the more advanced SCO recommendation generation algorithm, rather than STR, would have an impact on registry effectiveness. To this end, the registry was configured to use SCO, with all other aspects being identical to those of the previous experiment. The results are shown in Figure 9.5.

The use of the SCO algorithm has clearly had a significant impact on registry effectiveness. All 8 of the evaluated registry configurations have now generated recommendations that are an improvement over chance, having NCIOC values greater than 0. Moreover, relative to the previous experiment, the NCIOC values of all 8 configurations have been boosted. The values of the 3 highest-ranked configurations now range from 0.6867 to 0.7605. Indeed the 6 highest-ranked configurations have values which are greater than that of the top-ranked configuration in the previous experiment; the sixth-highest NCIOC value is 0.6317, compared with the top rank value of 0.5864 in Experiment One. Interestingly, the rankings of the configurations have also changed. For example, Location, ranked fourth in the last experiment, is now first; HourOfDay & Location, ranked first in the last experiment, is now second.

In terms of RSP, a consumer with tolerance level 1 had a 70.98% chance of finding an appropriate service with the top-ranked configuration (Location). At tolerance level 3, his chances had increased to 78.92%, and at level 5, to 79.78%. All of these values are distinctly higher than those of the top-ranked configuration in the previous experiment (33.96%, 66.34% and 75.72% respectively). Indeed, for these tolerance levels, the RSP



Situation Attribute Subset	Recommendations			Successful Recommendations Info			Cumul. IOC (level 5)	NCIOC (level 5)	Rank
	Success	Failure		Mean # Users	Mean # Services Selected	Mean # Service Selections			
		No service selections	No user situation						
No attributes	10397	0	0	868.54	64.96	39342.84	174.23	0.3648	7
Hour of day	10397	0	0	460.67	45.83	3257.31	170.49	0.357	8
Location	8737	1419	241	63.56	5.29	3449.91	363.23	0.7605	1
Role	10360	0	37	75	13.96	4865.22	315.16	0.6598	4
Hour of day & Location	8380	1776	241	42.7	2.83	324.84	345.51	0.7234	2
Hour of day & Role	10268	92	37	51.31	8.32	422.77	305.42	0.6395	5
Location & Role	7824	2332	241	64.32	4.1	3522.77	327.97	0.6867	3
Hour of day & Location & Role	7288	2868	241	46.07	2.62	349.51	301.72	0.6317	6

Figure 9.5: Experiment Two - What Impact Does SCO Have on Registry Effectiveness?

values of the second (HourOfDay & Location: 68.6%, 74.74% and 75.87%) and third-ranked configurations (Location & Role: 64.67%, 71.47% and 72.58%) are quite similar to those of the top-ranked configuration.

9.2.3 Experiment Three - What Impact does Time-Window Length Have on Registry Effectiveness?

The aim of the third experiment was to determine whether the length of time-window used would have an impact on registry effectiveness. The registry was configured to use the SCO recommendation algorithm, and the situation attribute subset of Location, as this partial configuration had produced the most effective recommendations so far (with a time-window length of 12 weeks). The registry was then configured to use, and evaluated for, each of the following lengths of time-window: 1 hour, 1 day, 1 week, 2 weeks, 4 weeks, 8 weeks, 12 weeks, and no time limit (i.e. as far back as the service selection history recorded). The results are shown in Figure 9.6.

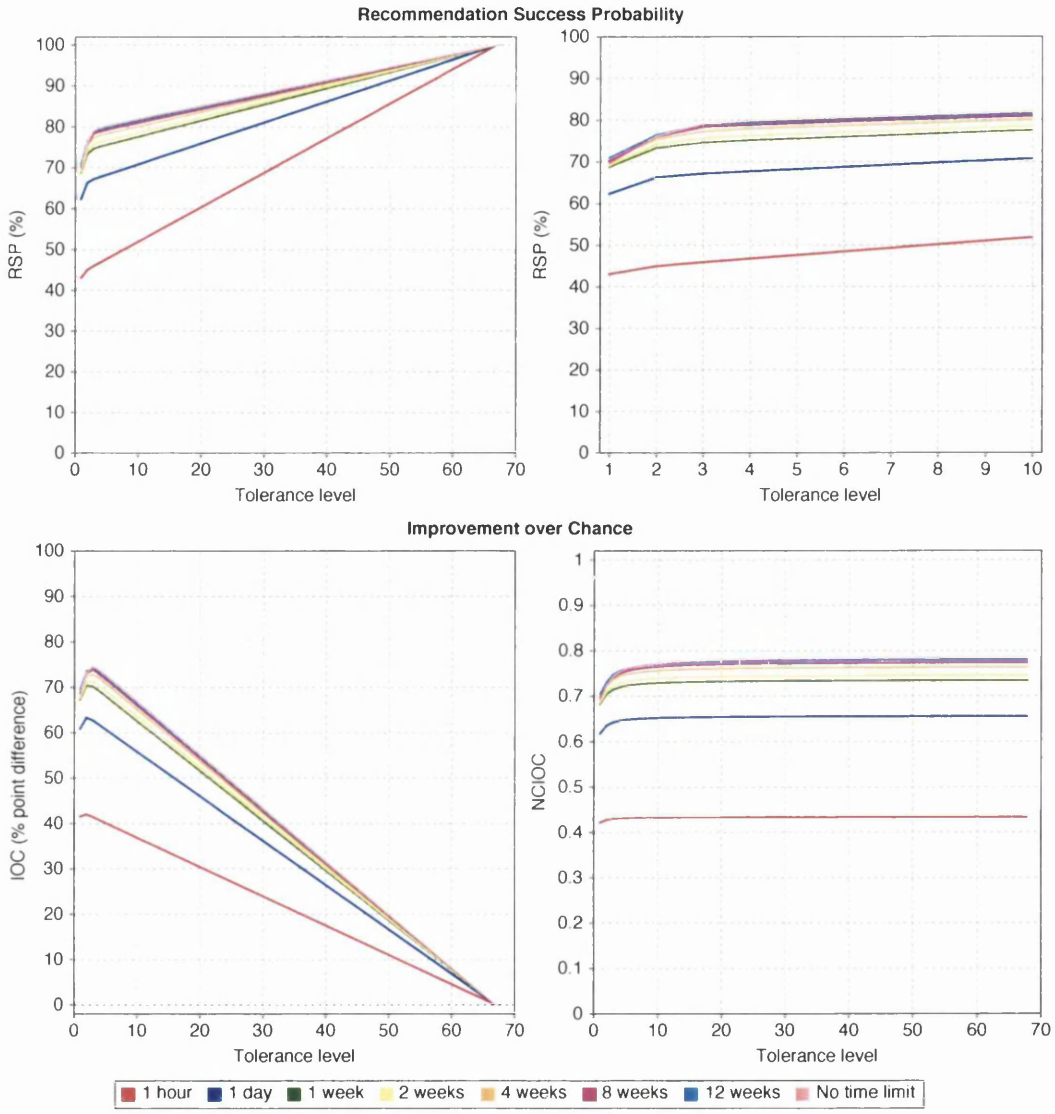
The largest NCIOC value occurred when the time-window was set to 12 weeks. For time-windows of less than 12 weeks, the smaller the time-window, the smaller the NCIOC value. For the time-window greater than 12 weeks ("no time limit"), the NCIOC value was also smaller. Presumably, for the ESSE set used in this evaluation, 12 weeks was the optimal "trade-off" length of time-window, as suggested in Section 8.2.4, that enabled the prototype registry to generate the most effective recommendations whilst reflecting relevant chance as rapidly as possible. Coincidentally, 12 weeks was the length of time-window I chose for the previous 2 experiments, based on my knowledge of the DCS printer scenario.

9.3 Conclusion

As was stated in Sections 8.1 and 9.1, the aim in constructing the prototype recommending registry and subsequently evaluating it was to ascertain whether the basic CF-based recommending registry design was both viable and valid.

Viability has been shown by the fact that it was possible to construct a working recommending registry which adhered to the basic design and did generate personalised service recommendations based on a recorded service selection history.

The validity of the basic design has been shown by the fact that the personalised service



Time-Window Length	Recommendations			Successful Recommendations Info			Cumul. IOC (level 5)	NCIOC (level 5)	Rank
	Success	Failure		Mean # Users	Mean # Services Selected	Mean # Service Selections			
		No service selections	No user situation						
1 hour	4886	5270	241	7.56	1.4	31.85	206.13	0.4316	8
1 day	7325	2831	241	19.8	1.99	130.12	309.37	0.6477	7
1 week	8181	1975	241	33.38	2.62	394.73	345.14	0.7226	6
2 weeks	8327	1829	241	36.29	2.83	512.57	349.89	0.7326	5
4 weeks	8593	1563	241	40.06	3.1	613.42	356.95	0.7474	4
8 weeks	8737	1419	241	59.38	4.2	2554	360.25	0.7543	3
12 weeks	8737	1419	241	63.56	5.29	3449.91	363.23	0.7605	1
No time limit	8737	1419	241	65.52	6.92	3776.49	362.43	0.7588	2

Figure 9.6: Experiment Three - What Impact does Time-Window Length Have on Registry Effectiveness?

recommendations generated by the prototype registry were effective, in that most registry configurations evaluated did produce recommendations which were a distinct improvement over chance, having NCIOC values significantly greater than 0. In particular, the top evaluated registry configuration (SCO algorithm, Location, 12 weeks; see Figure 9.5), with an NCIOC value of 0.7605, was just over three-quarters as effective as the perfect (and unattainable) recommending registry (i.e. NCIOC = 1). It should be noted that, had the prototype registry generated actual recommendations for actual uninformed consumers, it would probably have been even more effective. For the sake of stringency, the evaluation scheme disregards all past service selections of an ESSE selecting individual in generating an ESSE recommendation (see Section 7.4.2). Obviously, this would not happen in real registry deployment.

The results of the three experiments have also supported the hypothetical assumption that the effectiveness of recommendations generated by a CF-based recommending registry would be dependent on the three main integral aspects of the CF-based approach. The type-specific choices of situation attribute subset and length of time-window have a definite impact, which can be ascribed to the fact that these two particular aspects determine the relevant service selections identified on which a recommendation is based. The type of recommendation generation algorithm also has considerable impact, which is unsurprising, given that it dictates how the recommendation is actually generated from the relevant service selections. Of the two algorithms evaluated, SCO performed better than STR. Presumably, this can be attributed to the fact that SCO interprets the number of times a service was selected as a direct indication of its appropriateness, in contrast to the more basic interpretation of STR.

9.3.1 An Advanced Recommending Registry Design Justified

Given the demonstrated viability and validity of my basic CF-based recommending registry design, development of a more advanced version seemed justified. After consideration of the basic design in the light of the evaluation, I identified three design elements which could benefit from further investigation and improvement: the recommendation generation algorithm; the situation-similarity test used in identifying relevant service selections; and the developer task of registry configuration. Discussion of, and solutions to, these three aspects are presented in the following three chapters. A more advanced recommending registry design is subsequently defined in Chapter 13. This advanced design is an aug-

mented form of the basic CF-based design, which incorporates the improvements discussed and defined in the next three chapters.

9.4 Summary

The focus of this chapter has been the assessment of my basic CF-based recommending registry design in terms of viability and validity. This assessment was undertaken through the construction and evaluation of a design-adhering working prototype registry to recommend printers in the Department of Computing Science at the University of Glasgow.

The DCS printer scenario has been defined, and details of how the service selection history was recorded, and how the prototype registry was implemented, have been set out. Information has then been given about the evaluation setup for assessing prototype registry effectiveness, and the three evaluation experiments have been described and discussed.

The conclusion was drawn that the demonstrated viability and validity of the basic CF-based recommending registry design justified the design's development into a more advanced version. The next chapter is concerned with the first stage of this development process, namely research in connection with an advanced recommendation generation algorithm.

Chapter 10

An Advanced Recommendation Generation Algorithm

The focus of this chapter is the development of an alternative recommendation generation algorithm which would be more robust than SCO in the face of deliberate and devious manipulation of a design-adhering recommending registry. Such an algorithm, based on the notion of consensus, is defined, and various aspects of it are then discussed in greater detail. Finally, the consensus-based algorithm is evaluated under a number of different conditions.

10.1 The Problem of "Spamming"

The first element of my basic CF-based recommending registry design to be further developed was the recommendation generation algorithm. From the effectiveness evaluation results of the previous chapter, it might appear that there was little need to improve upon this design aspect. High levels of registry effectiveness (e.g. NCIOC = 0.7605) were demonstrated when the prototype was configured to use the SCO (Selection Count Ordering) recommendation generation algorithm. Surely SCO is a perfectly adequate choice for the recommending registry design? Unfortunately not: a recommending registry that used SCO could easily be manipulated by a service provider in order to achieve an undeservedly high recommendation rank for his service. In the face of such deliberate and devious manipulation, the effectiveness of the registry could be significantly diminished.

Imagine that a recommending registry is deployed in a commercial service-oriented scenario. In this competitive environment, every service provider wants to attract as

much custom as possible to his service. More users equate to more money, either through fees paid by the users themselves or through some form of advertising. Thus, there is a powerful financial motivation for a provider to ensure that his service achieves the highest rank possible in the recommendations generated by the registry, by fair means or foul. The higher the rank, the greater the possibility that the service will be selected and used by the requesting consumer, and more custom gained.

Imagine that a provider is dissatisfied with the ranks currently achieved by his service in the registry-generated recommendations, and wants to improve on them. How could he do this? Let us assume that the provider understands that the registry adheres to the CF-based approach, and knows how a personalised service recommendation is generated in response to a consumer's service request. More precisely, he knows that the consumer's recommendation is generated using an algorithm that ranks the available type-matching services by some measure of collective perceived appropriateness, through assessment of the relevant service selections identified from the service selection history. To recap, the relevant service selections are those which refer to a service of the requested type, and that were recently made in the consumer's situation; i.e. those that passed the three tests of type-matching, situation-similarity and recency defined in Section 8.3.1.

Given this information, the honest course of action that the service provider could take to achieve higher recommendation ranks would be to improve his service. For example, if the service was a grid DNA sequence-analyser in the running illustration of Bob, the provider might move the analyser to a more powerful machine with better network connections. This should lead to better processing-throughput of genome data, and better latency and bandwidth. The improvement should hopefully be noticed, and more people in more situations should begin selecting and using the service more often, perceiving it to be the most appropriate of the alternatives available. This change in service selection behaviour would be reflected in the relevant service selections identified by the registry, and should therefore translate into higher recommendation ranks for the provider's service. A higher rank would be deserved, given that the service had been improved.

The alternative, dishonest course of action that the service provider could take, rather than improving the service, would be to attempt to manipulate the process by which a recommendation was generated, so that the registry was *misled* into giving the service an undeservedly higher rank. Such deliberate and devious registry manipulation will be referred to as "spamming". This term is adopted from the field of web search, where it is

used to refer to any deliberate actions taken by a web page author in an attempt to mislead a search engine into giving his page an undeservedly high rank in a query result [51].

What form would this recommending registry spamming take? The ranking of services in a recommendation is determined by two elements: the relevant service selections identified from the service selection history, and the recommendation generation algorithm that assesses them. Thus, to spam the registry for a service s of type t in a particular situation p, the service provider (or a hired agent) would essentially have to make sham selections of s so that:

1. In the future, when the registry responded to a request made by a consumer in situation p for a service of type t, the sham selections would be identified as relevant. Thus, the sham selections would need to have been recently made in situation p.

2. When these identified relevant service selections, including the sham ones, were assessed by the recommendation generation algorithm used by the registry, s would be interpreted as being more appropriate than before spamming occurred, and would consequently be ranked higher in the type t recommendation returned to the consumer.

If the registry used the SCO recommendation generation algorithm, the provider could easily achieve a very high recommendation rank for his service through spamming. With SCO, a recommendation is generated by assessing the relevant service selections, and ranking the available type-matching services by the number of times each one was selected. Thus, in spamming the registry for service s in situation p, the provider could simply make repeated sham selections of s there. When the registry responded to a request made by a consumer in situation p for a service of type t, a large proportion of the relevant service selections identified would thus be sham service selections, and s should achieve a higher rank in the generated type t recommendation. Indeed, if over half the relevant service selections were sham selections, then s would be ranked in first place!

As an example, consider Figure 8.1 again. This figure illustrated a grid recommending registry responding to a request made by Bob in the 130.209.240 IP subnet (his situation) for a service of type Seq_Analyser, at 05/08/2004 11:27:32. Note that when SCO was used, Seq_Analyser D was ranked fourth in the generated recommendation, since it had not been selected at all. Imagine instead that the provider of D, Clara, had recently decided to spam the grid registry, being dissatisfied with the generally low recommendation ranks

that her analyser was achieving. She ascertains that the grid registry is using SCO, and that a generated Seq_Analyser recommendation is based on recent analyser selections that were made in the IP subnet of the requesting consumer. Having access to machines in certain subnets, Clara begins spamming the grid registry on 04/08/2004 by selecting and "using" D multiple times every day from each machine. One of the machines used by Clara is situated in the 130.209.240 IP subnet. Now consider Figure 10.1. This shows the grid recommending registry generating the same Seq_Analyser recommendation for Bob as in Figure 8.1, but in the face of Clara's spamming. Note that four of the relevant service selections now identified by the registry are sham selections of D made by Clara. Consequently, the grid registry generates a recommendation that ranks D in first place, since it was selected more times than any other analyser. Given that D would have been ranked fourth without Clara's deliberate manipulation, the spamming has clearly been a success.

The potential problem of devious manipulation of a recommending registry was initially alluded to in the original research question of how to generate a recommendation, detailed in Section 6.2.1.

10.2 The Need for an Alternative Recommendation Generation Algorithm

The evaluation results of the previous chapter do demonstrate that a recommending registry which used SCO could be effective, in terms of recommendations generated. However, in the DCS printer scenario in which the evaluation took place, the prototype registry was not being spammed. The registry was not actively deployed to generate printer recommendations to actual consumers, so logically no-one can have been attempting to boost the recommendation rank of a particular printer. Moreover, given that all the printers are owned by the same service provider (the Computing Science department), there is no reason why spamming should ever occur in this scenario.

However, in the face of spamming, the effectiveness of a recommending registry that used SCO could significantly diminish, as will be demonstrated later in the chapter. Given the ease with which the rank of a service could be boosted through the simple use of large numbers of sham service selections, the highest ranks of a registry-generated recommendation could very possibly be occupied by "spam" services. Since it is likely that most of

Bob's Service Request:
Service type: Seq_Analyser
Situation attributes: (NetworkLocation = "130.209.240")
Current date-time: 05/08/2004 11:27:32

Available Seq_Analyser Services: A, B, C, D, E, J

Relevant Service Selections

ServiceID	UserID	WhenOccurred
B	A64	23/07/2004 14:55:02
E	A62	27/07/2004 18:37:08
B	A29	27/07/2004 21:04:59
B	A29	02/08/2004 11:32:26
J	A12	02/08/2004 12:49:48
W	A09	03/08/2004 14:55:02
D	Clara	04/08/2004 10:00:00
D	Clara	04/08/2004 11:00:00
D	Clara	04/08/2004 12:00:00
D	Clara	04/08/2004 13:00:00
J	A62	05/08/2004 10:14:17

Seq_Analyser Recommendation:
1. D (would be ranked 4th without spamming)
2. B
3. J
4. E
5. A, C

Figure 10.1: Spamming the Grid Recommending Registry

these services would be inappropriate choices, the requesting consumer would be forced to search further to find an appropriate service, and the recommendation would be less effective.

As the SCO recommendation generation algorithm was considered an undesirable choice for a general recommending registry design, I decided to investigate and develop a more robust alternative algorithm that would enable a registry not only to be effective under normal conditions, but also to remain effective to a greater extent in the face of spamming. It is inevitable that a recommending registry that adhered to the CF-based approach could be affected by spamming in the form of sham service selections, and that its effectiveness could diminish. When a recommendation is generated, a sham selection looks no different to a normal one. However, as will be seen, it is possible to mitigate the effect that spamming has, through the use of a style of recommendation generation algorithm that limits the extent to which the recommendation rank of a spam service can be boosted.

10.2.1 A Different Style of Algorithm

In order to develop a different style of algorithm, I first reviewed the basic recommending registry design by considering the identified relevant service selections over which a recommendation generation algorithm operates, and the history-recorded form they take. As was noted in Section 8.2.6, the core attributes of a service selection history entry record the service selected (ServiceID), the individual responsible (UserID), and the date-time of occurrence (WhenOccurred). Viewed as a whole, therefore, the set of relevant service selections show a number of different individuals (each with a different UserID) selecting and using a variety of different type-matching services (each with a different ServiceID) over time. Under spamming conditions, the service selections made by a few of these individuals would be sham ones, explicitly made to mislead the recommending registry. However, more importantly, it can be assumed that those service selections made by all other individuals would *not* be sham ones. A non-spamming individual would have selected services which he did believe were most appropriate, without any ulterior motive.

With this last point in mind, I then developed a style of recommendation generation algorithm which should, through its particular assessment of the relevant service selections, be able to mitigate the effect of spamming individuals on the recommending registry:

> For each different selecting individual, infer his perceived opinion of the differ-

ent type-matching services in terms of appropriateness, through assessment of the particular relevant service selections made by him. Such an opinion will take the form of an appropriateness-ordered list of the type-matching services. Then, aggregate these individual opinions into a consensus opinion, which represents the "average" opinion of the *entire* group of individuals who made the relevant service selections. This consensus opinion, also in the form of an appropriateness-ordered list of type-matching services, will serve as the basis of the personalised service recommendation returned to the requesting consumer.

With this style of algorithm, the opinion of any individual will be absorbed into the group consensus opinion, and thus the extent to which a spam service is boosted in the resulting recommendation should be limited. Let us assume that only a small proportion of those individuals who made the relevant service selections were spamming the registry. In terms of the opinion of a spamming individual, inferred from the sham service selections made by him, his spam service might be highly ranked. However, in terms of the inferred opinions of non-spamming individuals, the spam service would probably be ranked much lower, with many other services being considered more appropriate. Thus, given the larger proportion of non-spamming individuals, the group consensus opinion should reflect a similar view, with the spam service being ranked below many of these other services.

Returning to the example of Clara in Figure 10.1, the relevant service selections identified by the grid registry in generating a Seq_Analyser recommendation for Bob were made by six different individuals. Since none of the five non-spamming individuals selected D, in contrast to Clara, this spam service should be ranked lowly in their inferred opinions, below the services that they did select (B, E, J and W). Thus, the resulting consensus-based recommendation returned to Bob should reflect a similar view, with D achieving only a low rank.

Obviously, if the majority of those individuals who made the relevant service selections were spamming the registry, then their opinions could not be adequately countered by those of the non-spamming minority, and spam services could achieve high ranks in the generated recommendation: garbage in, garbage out. However, this would seem an unlikely occurrence.

10.3 Social Choice Theory and Meta-Search

Owing to the novelty of my CF-based approach to personalised service recommendation, I was unable to find any pertinent research in either the area of service discovery, or that of CF, to aid me in the development of such a style of consensus-based recommendation generation algorithm. However, after further investigation, I did discover two other research areas, one in Economics and one in Computing Science, which did seem relevant. The two areas identified were Social Choice Theory and meta-search.

10.3.1 Social Choice Theory

Social Choice Theory is an area of Economics concerned with the study of electoral systems for making group decisions (i.e. a social choice). An election involves a number of "candidates" and a number of "voters". Typically, each voter expresses his opinion of the candidates by ranking them according to preference. These "preference rankings" are then aggregated to generate a consensus preference ranking, which represents the general opinion of the voters. Figure 10.2 provides an example of an election involving four candidates (named A to D) and five voters.

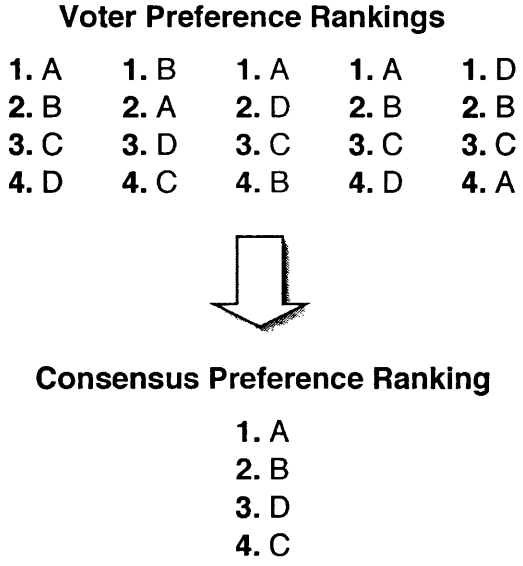


Figure 10.2: An Example Election

Elections of this form are used in various situations, either to choose a single "winning" candidate (the one that is top-ranked in the consensus preference ranking) or to obtain a

complete ranking of all the candidates. Perhaps the most well known type of election is a political one, in which the voting population of a geographic area choose one of a number of candidates for political office (e.g. an election for a member of parliament). However, elections are also used to choose a subset of candidates for a committee or board, and to rate the performance of competitors in judged sports such as diving, synchronised swimming, gymnastics and figure skating [112]. In the case of judged sports, a number of judges (voters) score the performance of each competitor (candidate), which results in preference rankings for all the judges that are then aggregated to determine the final performance ranking of the competitors.

Social Choice Theory is concerned with how voters' preference rankings are aggregated into a consensus preference ranking. Since the late eighteenth century, various electoral systems, or "rank aggregation methods", have been devised to perform this function. For example, such methods include Plurality Voting ("first past the post"), Approval Voting, Single Transferable Vote and Borda Voting [29]. Levin and Nalebuff [75] provide a description and analysis of seventeen different rank aggregation methods. However, despite the variations in these methods, almost all are based on the pairwise comparison of candidates. Essentially, the preference ranking of each voter can be interpreted as a set of pairwise comparisons (preferences). For example, in Figure 10.2, it can be inferred from the voter preference ranking of [A < B < C < D] (where "<" means ranked above) that A is preferable to B, A is preferable to C, A is preferable to D, B is preferable to C, B is preferable to D, and C is preferable to D. By tallying up all voters' pairwise comparisons for a particular pair of candidates, it is possible to determine which of the two candidates is generally preferred over the other. Typically, a rank aggregation method will apply this procedure to all pairs of candidates, and generate a consensus preference ranking by assessing the results in some way.

Referring back to the style of consensus-based recommendation generation algorithm outlined in Section 10.2.1, the generation of a personalised service recommendation can be seen to involve a form of election. In assessing the relevant service selections, the different selecting individuals can be viewed as "voters", and the different type-matching services selected as "candidates". Given that the inferred opinion of each individual is essentially phrased as a "preference ranking" of the type-matching services, with the services ranked by appropriateness, then theoretically a rank aggregation method from Social Choice Theory could be used to generate the consensus opinion that forms the basis of a personalised

service recommendation.

10.3.2 Meta-Search

Some research relating to the use of rank aggregation methods from Social Choice Theory has been undertaken in Computing Science, primarily in the Information Retrieval area of meta-search. A meta-search engine is a type of web search engine that does not respond to a user's information query by assessing some internal database of processed web pages; it does not have one. Rather, it submits the query to *other* search engines, which each return results in the typical manner, as a ranked list of web pages ordered by relevance. The meta-search engine then merges these multiple lists, and returns the resulting single list to the user. Proponents of meta-search argue that such a merged result list can be more consistent and more effective (in terms of precision and recall) than any of the individual result lists on which it is based [4, 98]. Certain meta-search researchers have realised that the merging of search engine results can also be viewed as a form of election. The underlying search engines queried can be viewed as "voters", and the ranked lists which they return as "preference rankings" of web page "candidates". Consequently, the use of certain rank aggregation methods from Social Choice Theory to generate the merged result list of web pages has been investigated [4, 35, 85, 98].

Of particular interest in connection with my research is the work of Dwork et al [35,36], who were motivated to use rank aggregation methods in meta-search to mitigate the effect of search engine spamming. As was noted earlier, "spamming" in the field of web search is used to refer to any deliberate actions taken by a web page author in an attempt to mislead a search engine into giving his page an undeservedly high rank in a query result. Dwork et al argued that the use of a rank aggregation method in a meta-search engine could counter any spamming that had successfully misled individual search engines whose results were being merged. That is, any spam pages that might have achieved high rankings in the results of some search engines would only achieve a low ranking in the "consensus" results of the meta-search engine. Dwork et al used a standard rank aggregation method in their research, and also developed some new methods to address spamming. Their results suggested that the use of these rank aggregation methods were indeed successful in mitigating the effects of search engine spamming. This research was thus of value to me when I was devising a consensus-based recommendation generation algorithm intended to mitigate the effects of recommending registry spamming.

10.4 The Consensus-Based Recommendation Generation Algorithm Defined

The consensus-based recommendation generation algorithm that I devised is defined below. The algorithm takes the form of a template, and consists of a sequence of four self-contained steps, each with a particular function. The first three steps can all be implemented in a number of different ways. Consequently, an overview of the entire algorithm will first be given, followed by a discussion of the particular variants of each step that were used in this research. The overview will be given with the aid of Figure 10.3, which provides a pictorial representation of the algorithm steps, and shows a recommending registry using the algorithm to generate a personalised service recommendation in response to an example service request.

10.4.1 Algorithm Overview

A proposed recommending registry responds to a consumer's service request by first identifying those type-matching services that are currently available. It also identifies relevant service selections from the service selection history (those which are type-matching, situation-similar and recent). The relevant service selections show a number of different individuals selecting and using a number of different type-matching services over time. Since the devised algorithm essentially takes the form of an electoral system in Social Choice Theory, these different selecting individuals will be referred to as "voters", and all the different services selected as "candidates", or "service candidates", as and when necessary.

In terms of the Figure 10.3 example, four available type-matching services have been identified: A, B, C and E. Nineteen relevant service selections have been identified, which show three different voters (Voter_1, Voter_2 and Voter_3) selecting and using four different service candidates (A, B, C and D). Note that a voter has not necessarily selected every service candidate (e.g. Voter_2 has not selected C or D).

Then, to generate a personalised service recommendation, a recommending registry must perform the following sequence of algorithm steps:

1. Calculate Voter Preference Rankings: For each voter, calculate his opinion of the service candidates in terms of appropriateness, through assessment of the particular rele-

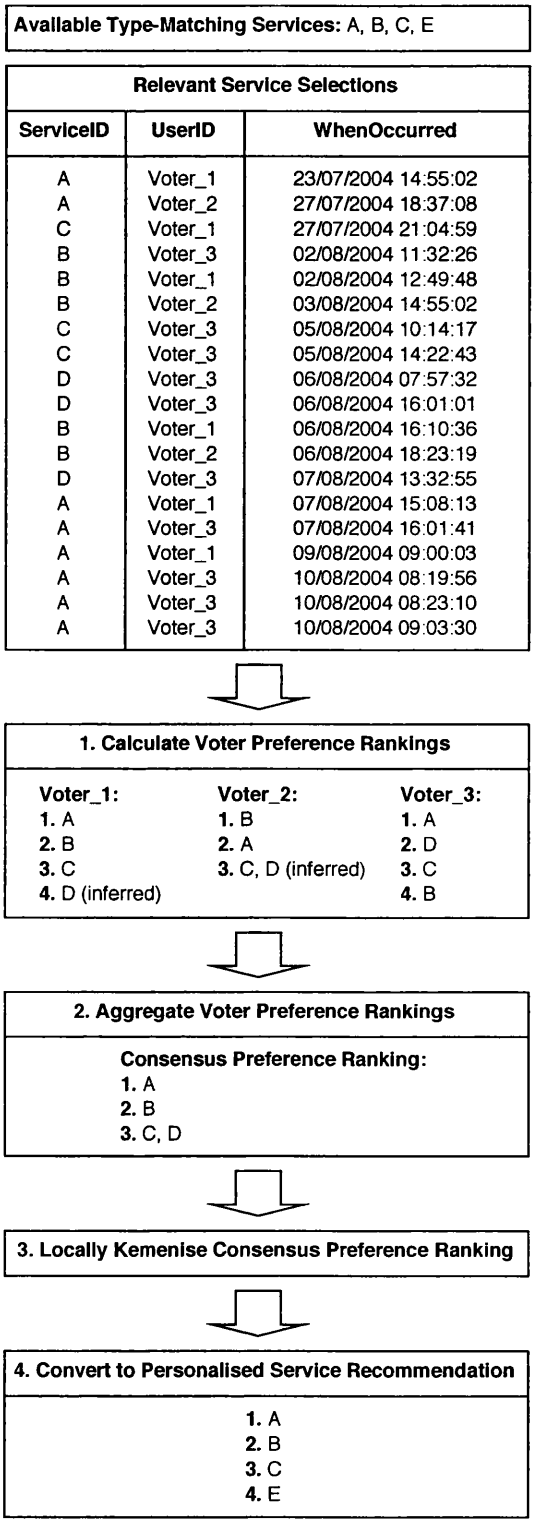


Figure 10.3: The Consensus-Based Recommendation Generation Algorithm

vant service selections made by him. This opinion will take the form of an appropriateness-ordered list of the service candidates, and will be referred to as a "preference ranking". If a voter has not selected certain service candidates, then it can be inferred that these are considered less appropriate than those that he did select. Thus, a complete preference ranking for such a voter can be obtained by placing the unselected service candidates in a bottom rank. A preference ranking may contain ties (i.e. be a partial order), as service candidates may be calculated to have equal appropriateness. The methods devised to calculate a voter preference ranking are defined in Section 10.4.2.

In terms of the Figure 10.3 example, for two out of the three voters (Voter_1 and Voter_2), a complete preference ranking of A, B, C and D has been obtained by placing unselected service candidates in a bottom rank (e.g. C and D for Voter_2).

2. Aggregate Voter Preference Rankings: Using a rank aggregation method, aggregate the voter preference rankings into a consensus preference ranking. The rank aggregation methods adapted and used in this research are defined in Section 10.4.3.

3. Locally Kemenise Consensus Preference Ranking: Apply a process known as "local Kemenisation" to the consensus preference ranking. For the sake of simplicity, this process is not detailed here, but in Section 10.4.4. However, to provide a brief explanation, local Kemenisation was developed by Dwork et al [35], and involves the partial reordering of a consensus preference ranking to ensure that it satisfies the "Extended Condorcet Criterion", an aspect of Social Choice Theory. In the context of meta-search, Dwork et al argued that local Kemenisation should further mitigate the effect of (search-engine) spamming on a consensus preference ranking of merged search-engine result lists. Theoretically, local Kemenisation should also be able to mitigate further the effect of recommending registry spamming on the consensus preference ranking generated in this algorithm.

4. Convert to Personalised Service Recommendation: Convert the locally Kemenised consensus preference ranking into a personalised service recommendation. The consensus preference ranking is an appropriateness-ordered list of the "candidate" type-matching services referred to by the relevant service selections. Thus, to convert it into a personalised service recommendation of available type-matching services, any candidate services that are not currently available must be removed. Moreover, any type-matching services that are available but are not candidates must be added. Given that these services

had not been selected by any of the voters, it can be inferred that they are generally considered less appropriate than the selected candidates, and can thus be placed in a bottom rank. The resulting appropriateness-ordered list of available type-matching services can then be returned to the requesting consumer as a personalised service recommendation.

In terms of the Figure 10.3 example, the consensus preference ranking contains one candidate service that is currently unavailable (D), and does not contain one of the available type-matching services (E). With D removed and E added, the consensus preference ranking is converted into a personalised service recommendation.

10.4.2 Algorithm Step One: Calculating a Voter Preference Ranking

I devised two methods, from first principles, to calculate the preference ranking of a voter from the relevant service selections made by him (algorithm step 1). One method is referred to as SCOVoter and the other as RPVoter.

SCOVoter

The SCOVoter (Selection Count Ordering Voter) method is based on the assumption that the number of times a voter selected a service is a direct indication of how appropriate he perceived it to be. Essentially more selections of a service are interpreted as more "votes of confidence" in its appropriateness. This assumption is very similar to that on which the original SCO recommendation generation algorithm was based. Thus, a voter's preference ranking is calculated by ordering the service candidates by the number of times each one was selected by him, from most to least. Unselected service candidates are placed in a bottom rank. The voter preference rankings in the Figure 10.3 example were calculated using the SCOVoter method. So, the preference ranking of [A < B < C < D] was calculated for Voter_1, as he selected the service candidates three, two, one and zero times respectively.

RPVoter

I devised the RPVoter (Reward Punishment Voter) method after further consideration of SCOVoter, in response to what I perceived as SCOVoter's inadequacies. Given that a personalised service recommendation is being generated for the "present", the preference ranking of a voter should ideally reflect his most recent opinion of service candidate appropriateness. However, a preference ranking calculated using the SCOVoter method

might not do this. For example, imagine that relevant service selections were identified from a large time-window, such as the last 6 months. A voter might have used service X a massive number of times in the first 5 months, and switched to using only service Y in the last month, considering it to be more appropriate. However, using SCOVoter, X would still be ranked above Y, as it had been selected a greater number of times overall.

A logical solution to this problem is to take into consideration the *sequence* in which a voter made his service selections, when calculating his preference ranking. If a service was only selected early in the sequence (i.e. not very recently), then it seems likely that the voter no longer considers it appropriate; services used later in the sequence seem likely to be considered more appropriate. Thus, I devised the RPVoter method to take into consideration the service selection sequence, and this method operates in the following manner. Please note that the description of method operation includes two variables - *new_service_offset* and *old_service_offset* - that are not explained until later. However, at this stage it is only necessary to know that these variables are positive integers predefined by the recommending registry developer.

Firstly, order the voter's relevant service selections by date-time, into the sequence in which they were made (oldest to most recent). Then, starting with the oldest service selection, process the ordered service selections one by one as defined below. For a processed service selection p which refers to a service s:

- If p is the first service selection in the sequence, then assign s a score of 0.
- If p is not the first service selection and s has not been selected before (i.e. s has not been referred to by an earlier service selection in the sequence) then:
    1. Identify the largest current score of all the services selected so far, *max_score*.
    2. Assign s a score of *max_score* − *new_service_offset*.
- If p is not the first service selection and s has been selected before then:
    1. Add 1 to the current score of s.
    2. Subtract 1 from the current scores of all other services selected so far.
    3. Identify the largest current score of all services selected so far, *max_score*.
    4. Let *min_acceptable_score* be *max_score* − *old_service_offset*. If the current score of s is less than *min_acceptable_score*, assign s a new score of *min_acceptable_score*.

Finally, having processed the entire sequence, rank the service candidates by their respective scores, from most to least. Unselected service candidates are placed in a bottom rank. The resulting ordered list is the voter's preference ranking.

Essentially, RPVoter involves processing a voter's relevant service selections in order of occurrence, rewarding each selected service (adding 1 point to its current score), and equally punishing all other services that have been selected before (subtracting 1 point from their current scores). This reflects the assumption that when the voter selects a particular service, he has decided that it is currently the most appropriate choice, whilst all the other services he has selected before are not. Thus, at any point during the processing of the voter's service selection sequence, by ordering the selected services by their current scores, it is possible to obtain the inferred preference ranking of the voter at that moment in time. The service with the largest score is considered most appropriate, and will be top-ranked. There are two "special cases".

Special Case One What initial score should be assigned to a service that has never been selected before? It can be assumed that the voter selected this "new" service either because he considered it to be most appropriate (as usual), or was trying it out. However, which interpretation is correct? Logically, if the voter proceeds to select the new service multiple times, then he does indeed consider it most appropriate, and it should ideally be top-ranked in the preference ranking as soon as possible. If he does not, and selects other services instead, then presumably he tried it out on a "one-off" basis, but it proved inappropriate. Assigning the new service a score of $max\_score - new\_service\_offset$ cautiously allows for both interpretations. Since *max_score* is the score of the currently top-ranked service, the new service will initially be ranked below it, as less appropriate. However, if the voter continues to select the new service, its score will increase, the score of the top-ranked service will decrease, and eventually the two services will switch ranking positions. The smaller the value of *new_service_offset* used, the less evidence required in the form of repeated voter service selections for the new service to be inferred as most appropriate and to achieve top rank. In contrast, if the new service is not selected again, its score will be eclipsed by those of the other selected services, and its rank in the preference ranking will decrease.

Special Case Two The second special case is somewhat similar to the first. When a processed service selection refers to a service that has been selected before, the score of

this service is increased by 1 and the scores of all other selected services decreased by 1. However, what if the selected service is "old", in the sense that it was last selected much earlier in the sequence, and therefore has a score much lower than that of the top-ranked service (*max_score*)? Having not selected the old service for so long, it can be assumed that the voter was either trying it out again, or did indeed now consider it to be most appropriate. Again, which interpretation is correct? As before, it would logically appear that the voter now considers the old service most appropriate if he proceeds to select it multiple times. If he does not, and selects other services instead, then presumably he tried it out on a "one-off" basis, but it again proved inappropriate. If the voter does consider the old service appropriate, it should ideally be top-ranked in the preference ordering as soon as possible. However, given its low score, the old service would have to be selected a large number of times to eclipse the score of the currently top-ranked service. The logical solution is to assign the old service a new score of *max_score* − *old_service_offset*, if its initial score is less than this value. Through this boost, the old service will be able to achieve the top rank faster, if the voter repeatedly selects it; the smaller the value of *old_service_offset*, the fewer selections required. However, if the old service is not selected again, this boosted score will be eclipsed by those of the other selected services, and its rank in the preference ranking will decrease.

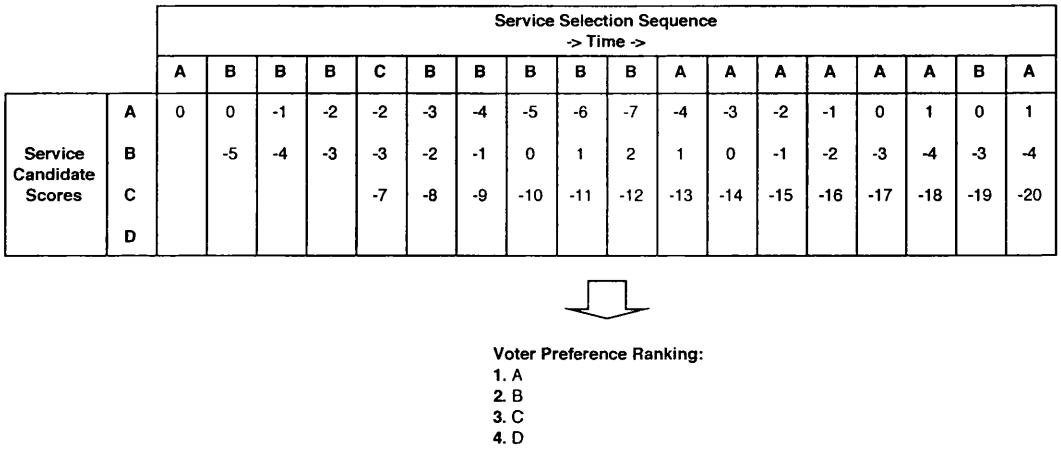


Figure 10.4: Using the RPVoter Method

Figure 10.4 provides an example of the RPVoter method being used to calculate a voter's preference ranking, with both *new_service_offset* and *old_service_offset* set to 5. There are four service candidates: A, B, C and D. The voter has made 18 relevant service

selections, which are ordered from left to right in the sequence in which they occurred (A, B, B, ... A, B, A). A, B and C were selected, but not D. After processing the entire service selection sequence, A has the highest score, followed by B and then C. Thus, the calculated preference ranking is [A < B < C < D], with unselected D being in the bottom rank. Note that if SCOVoter had been used, the calculated preference ranking would have been [B < A < C < D], even though B was used very little in the latter half of the sequence.

10.4.3 Algorithm Step Two: Aggregating Voter Preference Rankings

Five existing rank aggregation methods were adapted and used in this research to aggregate voters' preference rankings of the service candidates into a consensus preference ranking (algorithm step 2): Borda, and four methods based on Markov chains, known as MC_1, MC_2, MC_3 and MC_4. The Borda method was chosen as it is well-established in Social Choice Theory [75], is simple to implement, and has been used with some success in meta-searching [4]. The Markov chain methods were chosen as they were developed by Dwork et al [35] in their meta-search research into spamming mitigation, and therefore seemed of particular relevance.

To recap, in step 1 of the consensus-based recommendation generation algorithm, an appropriateness-ordered "preference ranking" of the "candidate" services (all those referred to by the relevant service selections) is calculated for each of the "voter" individuals responsible for the relevant service selections. In step 2, these voter preference rankings are then aggregated into a single consensus preference ranking of the service candidates.

In more formal terms, let S be the set of service candidates and V the set of voters. Let τ be the calculated preference ranking of a voter $v \in V$. τ is complete, in that it ranks all candidates in S, and may be a partial order (i.e. may contain ties). Let $R = \{\tau_1, \ldots, \tau_{|V|}\}$, the set of preference rankings for all voters in V. Then, in step 2, the rankings in R are aggregated into a single consensus ranking, $\hat{\tau}$.

Borda

In 1770, Jean-Charles de Borda proposed to the French Academy of Sciences the rank aggregation method that is now named after him [29]. The method assumes that each voter ranks every candidate in his preference ranking, which is a total order (i.e. contains no ties). The method operates as follows. For every voter preference ranking, a candidate is assigned a score equal to the number of candidates ranked below it. Thus, if there

are n candidates, the first-ranked candidate in a preference ranking is assigned a score of $n - 1$, the second-ranked candidate a score of $n - 2$, and so forth, with the bottom-ranked candidate being assigned a score of 0. The Borda score of each candidate is then obtained by totalling up its individual scores for the set of all voter preference rankings. A consensus preference ranking is calculated by ordering the candidates by Borda score, largest to smallest. Thus, the candidates are being ranked by the total number of pairwise comparisons that they "won" (i.e. those in which they were preferred to, and ranked above, the other candidate). The consensus preference ranking in Figure 10.2 was calculated using this method.

The Unfairness of Borda Despite its seeming applicability, the Borda method could not be used in the recommendation generation algorithm without minor modification. The voter preference rankings calculated in step 1 of the recommendation generation algorithm may be partial orders, whilst the original Borda method assumes only total orders. With Borda, a total of $\frac{n(n-1)}{2}$ points are essentially being allocated between the n candidates, for each voter's totally-ordered preference ranking. Thus, each voter contributes equally (in terms of points) towards the calculated consensus preference ranking. However, consider what would happen if some of the voters' preference rankings were partial orders, such as [A < (B C) < D] (where there are four candidates A, B, C and D). Using the original form of Borda, where a candidate is assigned a score equal to the number of candidates ranked below it, A would be assigned a score of 3 points, B and C a score of 1 point each, and D zero points; a total of 5 points. However, if a voter's preference ordering was a total order, the total number of points would be 6 (3 + 2 + 1 + 0)! Thus, the different voters would contribute unequally towards the calculated consensus preference ranking, which would be unfair. My slight modification of Borda was designed to address this unfairness.

My Adaptation of Borda The modification of Borda I devised is based on the assumption that, even if the preference ranking of a voter calculated in step 1 of the recommendation generation algorithm is a partial order, the voter's true opinion is actually a *total* order. Essentially, if only a partial order was calculated, there was not enough information in the voter's relevant service selections to determine his true preference ordering of rank-tied candidates. From this perspective, a partially-ordered preference ranking can be interpreted as representing the set of totally-ordered preference rankings that are obtained by breaking rank ties in every possible way. One of these total orders is assumed to

be the voter's true preference ranking, with equal probability. For example, reconsidering [A < (B C) < D], the two corresponding total orders are [A < B < C < D] and [A < C < B < D], with each having a 0.5 chance of being the true preference ranking.

Based on this interpretation, my modified Borda method calculates a candidate's score for a voter's partially-ordered preference ranking as the *average* of the scores that would be assigned to it in the corresponding totally-ordered preference rankings. For example, for [A < (B C) < D], B would be assigned a score of 1.5 (as would C), since 2 candidates are ranked below it in one of the totally-ordered rankings, and 1 candidate in the other. The total of points allocated to the candidates in a partially-ordered voter preference ranking (6 in the example) is the same as that in a totally-ordered voter preference ranking, so the modified Borda method is fair. As normal, the Borda score of each candidate is obtained by totalling up its individual scores for the set of all voter preference rankings, and a consensus preference ranking calculated by ordering the candidates by this score. The consensus preference ranking in Figure 10.3 was calculated using this modified method.

More formally, the modified Borda method can be defined as follows. For each service candidate $s \in S$ and voter preference ranking $\tau \in R$, calculate $b(s, \tau)$, the average number of candidates ranked below s in the total orders corresponding to τ[1]. In defining $b(s, \tau)$, let:

- r be the rank of τ occupied by s.
- $|r|$ be the number of candidates in rank r.
- a be the number of candidates in ranks above r.

Then:

$$b(s, \tau) = |S| - (a + \frac{|r| + 1}{2}) \tag{10.1}$$

Continuing, for each candidate $s \in S$, calculate the Borda score $B(s, R)$, the sum of $b(s, \tau)$ for all $\tau \in R$. That is:

$$B(s, R) = \sum_{\tau \in R} b(s, \tau) \tag{10.2}$$

Finally, to calculate the consensus preference ranking $\hat{\tau}$, order all $s \in S$ according to $B(s, R)$, largest to smallest.

[1] A totally-ordered preference ranking simply corresponds to a single total order: itself.

Markov Chain Methods: MC_1 - MC_4

The four rank aggregation methods of MC_1, MC_2, MC_3 and MC_4 developed by Dwork et al [35] for meta-search are all variants of an approach based on the use of a Markov chain. As Dwork et al state, "a (homogeneous) *Markov chain* for a system is specified by a set of states $S = \{1, 2, \ldots, n\}$ and an $n \times n$ non-negative, stochastic (i.e., the sum of each row is 1) matrix M. The system begins in some start state in S and at each step moves from one state to another state. This transition is guided by M: at each step, if the system is in state i, it moves to state j with probability M_{ij}. If the current state is given as a probability distribution, the probability distribution of the next state is given by the product of the vector representing the current state distribution and M. In general, the start state of the system is chosen according to some distribution x (usually, the uniform distribution) on S. After t steps, the state of the system is distributed according to xM^t. Under some niceness conditions on the Markov chain (whose details we will not discuss), irrespective of the start distribution x, the system eventually reaches a unique fixed point where the state distribution does not change. This distribution is called the *stationary distribution.* It can be shown that the stationary distribution is given by the principal left eigenvector y of M, i.e., $yM = \lambda y$. In practice, a simple power-iteration algorithm can quickly obtain a reasonable approximation to y.".

Dwork et al maintain that the probabilities in the stationary distribution vector y define a natural ordering on S. That is, the states in S can be ranked according to their stationary probability, largest to smallest. Dwork et al refer to this ordering as the *Markov chain ordering of M.* They propose a style of rank aggregation method in which every candidate in the election corresponds to a state in S, with the transition probabilities in M depending in some particular way on the voters' preference rankings of the candidates, and the resulting Markov chain ordering of M being the consensus preference ranking. Essentially, the transition probabilities specified in M correspond to a probabilistic switch from the current "candidate" state to a "better candidate" state. Thus, the larger the stationary probability of a candidate state, the "better" the candidate is calculated to be.

Dwork et al argue that there are several motivations for using a Markov chain as the basis of a rank aggregation method [35]. For example, it is argued that such a method could handle incomplete preference rankings, when not all voters had ranked (i.e. compared) all candidates. The claim is made that, having used available comparisons between pairs of candidates i and j to determine the transition probability between i and j, the connectivity

of the Markov chain could be exploited to (transitively) "deduce" comparison outcomes between candidate pairs that were not explicitly ranked by any of the voters. Secondly, it is stated that a consensus preference ranking could be calculated efficiently using a Markov chain. By explicitly computing the transition matrix M in $O(n^2k)$ time (where n is the number of candidates and k the number of voters), an approximation of the stationary distribution can be obtained with a few iterations of the Power method [9]. Dwork et al suggest that it is actually possible to identify the top few candidates in the stationary distribution in $O(nk)$ time, after some preprocessing.

The four proposed variants of this style of rank aggregation method, MC_1, MC_2, MC_3 and MC_4, differ in the specification of the Markov chain transition matrix M. Dwork et al provide a justification of the specifications in [35]. The different specifications are given below. For all four methods, the assumption is made that voters' preference rankings could be incomplete, not containing (ranking) all the candidates, and could be partial orders.

MC_1: If the current state is candidate i, then the next state is chosen uniformly from the multiset of all candidates that were ranked higher than or equal to i in some preference ranking that contained i.

MC_2: If the current state is candidate i, then the next state is chosen by first uniformly picking a preference ranking τ from all those that contained i, then uniformly picking a candidate from all those candidates that were ranked higher than or equal to i in τ.

MC_3: If the current state is candidate i, then the next state is chosen as follows: first pick a preference ranking τ uniformly from all those that contained i, then uniformly pick a candidate j that was ranked by τ. If j was ranked higher than i in τ, then go to j, else stay in i.

MC_4: If the current state is candidate i, then the next state is chosen as follows: first uniformly pick a candidate j from the set of all candidates. If j is ranked higher than i by the *majority* of preference rankings that contained both i and j, then go to j, else stay in i.

My Use of the Markov Chain Methods It was possible to use all four rank aggregation methods in the recommendation generation algorithm without modification. Unlike

the original version of Borda, the methods can process the complete and possibly partially-ordered voter preference rankings R calculated in step 1 of the algorithm. However, in this research, I decided to experiment with two versions of each method. Dwork et al asserted that a rank aggregation method based on a Markov chain (MC) could handle incomplete voter preference rankings. In step 1 of my recommendation generation algorithm, the initial preference ranking of service candidates calculated for a voter could indeed be incomplete, if the voter had not selected certain candidates (see Section 10.4.1). As has been stated, if this occurred, an "inferred" complete ranking would be obtained by placing these unselected candidates in a bottom rank, on the assumption that they were considered less appropriate by the voter than those that he did select. Given that an MC-based rank aggregation method could handle either the initial and possibly incomplete voter preference rankings, or the inferred complete rankings, I decided to investigate each approach.

Two versions of each method were devised, which differ in their intepretation of R, the set of complete voter preference rankings calculated in algorithm step 1. In calculating a consensus preference ranking, the first, standard version ignores any inferred bottom ranks of voter preference rankings, only considering the rankings in their initial and possibly incomplete form. This version is referred to by the method's original name (e.g. MC_1). In contrast, the second, alternative version considers the rankings in their (possibly inferred) complete form. This version is referred to by the method's original name, prefixed by "Inf" (which represents "Inferring"; e.g. InfMC_1).

However, even on initial assessment, it would appear that, conceptually, the "non-Inf" version of any MC-based rank aggregation method has certain undesirable qualities for a proposed recommending registry. Primarily, it is unfair, in that a voter with an incomplete preference ranking would contribute less towards the calculated consensus preference ranking than someone with a complete ranking. The transition probabilities for candidate i in matrix M (i.e. all M_{ij}, where j is another candidate) are based only on voter preference rankings that contain i. Thus, a voter with an incomplete preference ranking that does not contain i would not contribute towards these probabilities. More importantly, if fewer voters contribute towards these probabilities, the easier it would be for one of these contributing individuals to spam the registry successfully for i. The fewer the voters, the greater the spammer's contribution on the M_{ij} transition probabilities, and the higher the rank potentially achieved by the spam candidate i in the calculated consensus preference ranking (and thus the recommendation). In contrast, my alternative "Inf" version of any

MC-based rank aggregation method is fair. Since all voters will contribute (equally) towards every single transition probability in M (every voter ranks every candidate), the impact of any spamming voter should be mitigated as much as possible. For similar reasons, the impact of any non-spamming voter with idiosyncratic behaviour, who ranked generally inappropriate services highly, should also be mitigated.

Example Usage of MC-based Methods in My Consensus-Based Recommendation Generation Algorithm Figure 10.5 provides an example[2] of the MC-based rank aggregation methods being used to calculate a consensus preference ranking in the recommendation generation algorithm. For the three voter preference rankings $R = \{\tau_1, \tau_2, \tau_3\}$ of the three service candidates $S = \{A, B, C\}$ specified at the top of the figure, it shows the corresponding transition matrices determined for InfMC_1, InfMC_2, InfMC_3 and InfMC_4: M^1, M^2, M^3 and M^4 respectively.

Rank	τ_1	τ_2	τ_3
1	A	C	C
2	B	A	B
3	C	B	A

M^1

Candidate	A	B	C
A	3/6	1/6	2/6
B	2/7	3/7	2/7
C	1/5	1/5	3/5

M^2

Candidate	A	B	C
A	11/18	2/18	5/18
B	5/18	8/18	5/18
C	2/18	2/18	14/18

M^3

Candidate	A	B	C
A	6/9	1/9	2/9
B	2/9	5/9	2/9
C	1/9	1/9	7/9

M^4

Candidate	A	B	C
A	2/3	0	1/3
B	1/3	1/3	1/3
C	0	0	1

Figure 10.5: Example Transition Matrices

[2]This example is adapted from one given by Renda and Straccia [98]

Some example computations for the matrix entries will now be given. Remember that M_{ij}^k is the probability that, if the current system state is candidate i, then the state after the next transition is candidate j.

- M_{AC}^1 is 2/6. The multiset H of candidates ranked higher than or equal to A in some preference ranking that contained A is $\{A, A, C, A, B, C\}$. Thus, the probability of uniformly choosing one of the candidates in H is 1/6, and the probability of choosing C is 2/6.

- M_{BA}^2 is 5/18. The probability of uniformly choosing a preference ranking τ from all those containing B is 1/3. For a preference ranking τ_k containing B, let H_{B,τ_k} be the set of all candidates ranked higher than or equal to B. If τ_1 is selected, then $H_{B,\tau_1} = \{B, A\}$, and the probability of choosing A is 1/2. Similarly, $H_{B,\tau_2} = \{B, A, C\}$, and $H_{B,\tau_3} = \{B, C\}$. Thus, $M_{BA}^2 = (\frac{1}{3} * \frac{1}{2}) + (\frac{1}{3} * \frac{1}{3}) + (\frac{1}{3} * 0) = \frac{5}{18}$.

- M_{BC}^3 is 2/9. The probability of uniformly choosing a preference ranking from all those containing B is 1/3. The probability of uniformly choosing a candidate from such a preference ranking is 1/3 as well. Comparing C and B in each preference ranking τ containing B, C is not ranked higher than B in τ_1, but is in τ_2 and τ_3. Thus, $M_{BC}^3 = (\frac{1}{3} * 0) + (\frac{1}{3} * \frac{1}{3}) + (\frac{1}{3} * \frac{1}{3}) = \frac{2}{9}$.

- M_{BB}^4 is 1/3. The probability of uniformly choosing a candidate $s \in S$ is 1/3. Additionally, consider the following table:

Candidate	A	B	C
A	0	1/3	2/3
B	2/3	0	2/3
C	1/3	1/3	0

  An entry e_{ij} in the table is the proportion of preference rankings containing both candidates i and j that ranked j above i. If $e_{ij} > 1/2$, then the *majority* ranked j above i. M_{BB}^4 is the probability that, if the current system state is candidate B, then the system remains in the same state B after the transition. As e_{BA}, e_{BB} and e_{BC} are 2/3, 0 and 2/3 respectively, the system moves away from candidate B in two cases out of three, each with probability 1/3. Thus, M_{BB}^4 is the remaining 1/3.

The consensus preference ranking $\hat{\tau}$ of the voter preference rankings R is the Markov chain

ordering on M^k, $k = 1 \ldots 4$. It can be shown that, for all four cases (InfMC_1 - InfMC_4), $\hat{\tau} = [C < A < B]$.

10.4.4 Algorithm Step Three: Local Kemenisation

The third step in the consensus-based recommendation generation algorithm involves the consensus preference ranking calculated in step 2 being partially reordered according to a process of "local Kemenisation" developed by Dwork et al [35].

Background: the Extended Condorcet Criterion To understand the concept of local Kemenisation, it is first necessary to understand an aspect of Social Choice Theory known as the "Condorcet Criterion", and its natural extensions. The Condorcet Criterion [121] states that, in an election in which every voter ranks every candidate, if a particular candidate is ranked above all others by an absolute majority of voters, it should be declared the winner. That is, in the consensus preference ranking, this candidate should be top-ranked. A natural extension, attributed to Truchon [111], states that if a candidate is ranked above another by an absolute majority of voters, it should be ranked above this other candidate in the consensus preference ranking. This is called the "Extended Condorcet Criterion" (ECC).

Dwork et al identified ECC as being of relevance in addressing search engine spamming in a meta-search engine. They were concerned with merging the web-page result lists returned by a set of queried search engines into a single result list, using rank aggregation methods. The merging procedure was viewed as an election, with the queried search engines as voters, and the top d elements of their returned result lists (such as the top 100) as preference rankings of web-page candidates. These "top d" result lists are unlikely to contain exactly the same web-pages, given that different search-engines use different ranking functions and may index different portions of the World Wide Web. Thus, the total set of candidates was taken to be the union of web-pages ranked in the considered top d result lists, with the lists thus being interpreted as incomplete preference rankings. The MC-based rank aggregation methods were specifically developed to aggregate these incomplete preference rankings into a complete consensus preference ranking of the web-page candidates (the merged result list).

Dwork et al defined a relaxed version of the Extended Condorcet Criterion that was applicable to their meta-search arrangement of incomplete voter preference rankings (normal

ECC assumes completeness): if a candidate i is ranked above candidate j by the majority of voters *who ranked both candidates*, then i (the Condorcet winner) should be ranked above j (the Condorcet loser) in the consensus preference ranking. This relaxed version will be referred to as ECC^D. Dwork et al argued that, by ensuring that the consensus preference ranking (of web-page candidates) calculated by a meta-search engine satisfied ECC^D, the effect of search engine spamming could be mitigated.

Imagine that, for a particular query, a spam web-page s has achieved an undeservedly high rank in some of the search engine preference rankings aggregated by the meta-search engine. Interpreting one of these successfully spammed preference rankings as a set of pairwise comparisons, s is undeservedly preferred to (i.e. ranked above) a number of "better" web-pages. Let us consider one of these pairs, s and a better page b, for all search engines that ranked both web-pages in their preference rankings. If the majority of these search engines have been successfully spammed for this pair of web-pages, with s being preferred to b, then the rank aggregation method is working with overly bad data, and nothing can be done to mitigate the effect of such spamming. However, if more than half of these search engines still prefer b to s, then b is the Condorcet winner and s the Condorcet loser. Thus, in a consensus preference ranking that satisfies ECC^D, better web-page b will be ranked above spam web-page s. Given that, in such a ranking, the Condorcet winner of *every* pair of (web-page) candidates will be ranked above the corresponding Condorcet loser, spam web-pages such as s that are predominately Condorcet losers should only achieve low ranks, and the effect of search engine spamming should be mitigated to a significant extent.

Dwork et al noted that many of the existing rank aggregation methods, including Borda and the MC-based approaches, do not always generate consensus preference rankings that satisfy ECC^D. Thus, they developed the method of "local Kemenisation", which can be used to partially reorder any initial consensus preference ranking so that it *does* satisfy ECC^D, and therefore gains the associated "anti-spamming" benefits. The method is described below; in the description, the terms "consensus preference ranking" and "aggregation" (of voter preference rankings) will be used interchangeably.

The Method Defined Local Kemenisation is inspired by a proposal made by Kemeny [63] for the "ideal" consensus preference ranking. Kemeny's proposal is the following: given a set of k complete totally-ordered voter preference rankings $\tau_1, \tau_2, \ldots, \tau_k$

of n candidates, produce the complete totally-ordered ranking σ of the candidates that minimises $\sum_{i=1}^{k} K(\tau_k, \sigma)$, where $K(\tau, \sigma)$ (the K-distance) is the number of candidate pairs (i,j) on which the rankings τ and σ disagree (one of them ranks i above j, whilst the other ranks j above i). σ is called the "Kemeny optimal aggregation", and of all possible aggregations, is the one that disagrees least with the voter preference rankings in terms of pairwise comparisons. Most importantly, σ satisfies ECC^D. However, computing a Kemeny optimal aggregation is NP-hard, even when there are only four voters.

Dwork et al introduced the related notion of a *locally* Kemeny optimal aggregation, which is locally rather than globally optimal in terms of minimising the total number of disagreeing pairwise comparisons, but still satisfies ECC^D, and *is* computationally tractable. Since they were developing the concept in the context of meta-search, Dwork et al assumed that voters' preference rankings could be incomplete, and that the set of candidates was the union of elements in these rankings. Thus, given a set of k (possibly incomplete) totally-ordered voter preference rankings $\{\tau_1, \tau_2, \ldots, \tau_k\}$ of n candidates, a complete totally-ordered ranking π of the candidates is a locally Kemeny optimal aggregation with respect to $\{\tau_1, \tau_2, \ldots, \tau_k\}$ if there is no ranking π' that can be obtained from π by performing a single transposition of an adjacent pair of candidates and for which $\sum_{i=1}^{k} K(\tau_k, \pi') < \sum_{i=1}^{k} K(\tau_k, \pi)$. In other words, π is locally Kemeny optimal if it is not possible to reduce the total K-distance to the voter preference rankings (the total number of disagreeing pairwise comparisons) by flipping an adjacent pair in π, such as the first and second ranked candidates. A locally Kemeny optimal aggregation can be shown to satisfy ECC^D, and can be computed in $O(kn \log n)$ time.

The process of local Kemenisation developed by Dwork et al takes an initial totally-ordered aggregation μ of (possibly incomplete) totally-ordered voter preference rankings $\{\tau_1, \tau_2, \ldots, \tau_k\}$, and computes a locally Kemeny optimal aggregation π with respect to $\{\tau_1, \tau_2, \ldots, \tau_k\}$ that is maximally consistent with μ. That is, π only disagrees with μ on the order of any given pair of candidates if a majority of τ's who expressed an opinion (i.e. ranked both candidates) disagreed with μ. Thus, π can be viewed as the "tweaked" version of μ that satisfies ECC^D with respect to $\{\tau_1, \tau_2, \ldots, \tau_k\}$. The local Kemenisation process is actually quite simple. It involves π being constructed incrementally, with the n^{th} ranked candidate in μ being inserted into an intermediate version of π that contains all $n-1$ candidates ranked above it in μ, which have previously been processed. Let the n^{th} (ranked) candidate in μ being processed be x. Then x is inserted into π just below the

lowest-ranked candidate y such that (a) the majority of τ's who expressed an opinion did *not* prefer x to y; and (b) for every successor z of y in π, the majority of τ's who expressed an opinion *did* prefer x to z. In other words, x is inserted at the bottom of the ranking π, and is "bubbled" up the ranking as long as it is consistent with the majority opinion of the τ's. Once all the candidates in μ have been processed, π is the locally Kemeny optimal aggregation with respect to $\{\tau_1, \tau_2, \ldots, \tau_k\}$ that is maximally consistent with μ.

My Use of Local Kemenisation The argument made by Dwork et al concerning the anti-spamming benefits of ECC^D is as relevant to a proposed CF-based recommending registry as it is to a meta-search engine. The consensus-based recommendation generation algorithm may involve the aggregation of voter preference rankings of services rather than web-pages, but the problem of spamming is exactly the same. As long as only a minority of voters rank spam services highly, many spam services should equate to Condorcet losers, and should only achieve low ranks in an ECC^D-satisfying consensus preference ranking (and thus the corresponding personalised service recommendation). Consequently, I decided that in step 3 of the recommendation generation algorithm, the consensus preference ranking $\hat{\tau}$ of service candidates S calculated from the voter preference rankings R in step 2 should be locally Kemenised with respect to R, to ensure that it satisfied ECC^D.

It was possible to use the process of local Kemenisation in my recommendation generation algorithm without modification, although if $\hat{\tau}$ contained ties, these would need to be broken; the processed aggregation is required to be a total order. Dwork et al also assumed that the voter preference rankings would be total orders, but some of the rankings R calculated in step 1 of the recommendation generation algorithm could be partial orders. However, given that local Kemenisation simply involves the individual comparison of candidate pairs within a voter preference ranking to determine whether one candidate is ranked above the other, this should have no impact.

Dwork et al developed the local Kemenisation process so that it could compute a locally Kemeny optimal aggregation with respect to incomplete voter preference rankings. Thus, as with the MC-based rank aggregation methods for the recommendation generation algorithm, two versions of the local Kemenisation process were devised which differ in their interpretation of the complete voter preference rankings R of the service candidates. In locally Kemenising $\hat{\tau}$, the first, standard version ignores any inferred bottom ranks of voter preference rankings (which contain unselected service candidates), only considering

the rankings in their initial and possibly incomplete form. This version is referred to as "LK". In contrast, the second, alternative version considers the rankings in their (possibly inferred) complete form. This version is referred to as "InfLK" (as in "Inferring" local Kemenisation).

Even on initial assessment, it would appear that, as with the "non-Inf" versions of MC-based rank aggregation methods, the "non-Inf" version of local Kemenisation (LK) has certain undesirable properties for a proposed recommending registry. Essentially, consideration of voters' preference rankings only in their initial and possibly incomplete form could actually make it *easier* for a spam service to achieve a high rank in a locally Kemenised consensus preference ranking. Imagine that $\hat{\tau}$ is being locally Kemenised using LK, and that spam service s is top-ranked in a single (complete) voter preference ranking r in R. Being highly inappropriate, s has not been selected by any other voter, so LK considers all the other preference rankings in R to be incomplete, not containing s. Consider what happens to s during the local Kemenisation process. Let $\hat{\pi}$ be the locally Kemenised form of $\hat{\tau}$. As normal, s is inserted into the intermediate version of $\hat{\pi}$ at the lowest rank where, for every candidate c ranked below, the majority of voter preference rankings in R that contained both candidates preferred s to c. Using LK, the only voter preference ranking taken into consideration during the insertion of s is r, as it is the only one that contained the spam service. Thus, since s is top-ranked in r, it is preferred to every other candidate by the majority (of one!), and will be inserted at the top rank in $\hat{\pi}$. Clearly, LK may not always mitigate the effect of spamming. In contrast, s would not achieve such a high rank in $\hat{\pi}$ using InfLK. Since every voter preference ranking in R would be taken into consideration during the insertion of s (or any other candidate), and s is ranked lowly by the absolute majority, it would only achieve a correspondingly low rank in $\hat{\pi}$. It should be noted that, rather than being a spam service, s could be an inappropriate service ranked highly by a non-spamming voter with idiosyncratic behaviour, but the effect would be the same.

10.5 Evaluation of the Consensus-Based Recommendation Generation Algorithm

To determine whether, compared to the original recommendation generation algorithm of SCO, the consensus-based recommendation generation algorithm did indeed enable

a recommending registry not only to be effective under normal conditions, but also to remain effective to a greater extent in the face of spamming, a number of experiments were performed. The experiments take a similar form to those of the last chapter, with different configurations of the prototype recommending registry being evaluated for effectiveness under both normal and simulated spamming conditions using the ESSE-based scheme of Chapter 7. The same DCS printer scenario ESSE set was used, consisting of the 10397 printer selections identified between 04/01/2004 00:00 and 31/01/2004 00:00 from the recorded departmental service selection history, with the ESSE condition defined as $m = 5$ and $n = 28$ days. Thus, the results of the following experiments can be compared against those of the previous chapter.

The consensus-based recommendation generation algorithm was implemented as a configurable version of the Recommendation Generator component in the registry prototype (see Figure 9.2). The different variants of each algorithm step (step 1: SCOVoter and RPVoter; step 2: Borda, MC_1, $InfMC_1$, MC_2, $InfMC_2$, MC_3, $InfMC_3$, MC_4 and $InfMC_4$; step 3: LK and InfLK) were all implemented, so that the registry could be configured to use any variation of the algorithm.

In all the experiments that follow, the prototype registry was configured to use a time-window of 12 weeks. The primary aspect of investigation was the recommendation generation algorithm, so the time-window length was fixed to a constant; 12 weeks was chosen since the registry had proved most effective when configured to use this value. However, for each experiment, it was decided to vary the situation attribute subset used to assess service selection situation-similarity. To recap, there are 8 possible subsets of the 3 printer-specific situation attributes of HourOfDay, Location and Role. The particular subset chosen significantly affects the specific relevant service selections identified and the consequent recommendations generated. Thus, it seemed worthwhile to investigate the consensus-based recommendation generation algorithm (henceforth referred to as "CB" for brevity) for all different situation attribute subsets; if certain behaviour was noted across most subsets, it seemed likely to be a general trend. After consideration, it was also decided that, when the registry was configured to use a CB variation, algorithm step 1 would always be set to RPVoter, with *new_service_offset* = 5 and *old_service_offset* = 5. This made evaluation more tractable, as it halved the possible number of CB variations that could be investigated (since the less desirable SCOVoter was disregarded). The offsets were set to 5, as this ensured that a newly selected departmental printer (or an old one) could

only achieve top rank in a voter's preference ranking after at least 4 repeated selections. If a voter had repeatedly selected and walked to a printer 4 times, it seemed likely that he now considered it most appropriate. Further investigation of SCOVoter and optimum RPVoter offsets could be undertaken in the future.

10.5.1 Experiment One - CB Under Normal, Non-Spamming Conditions

The aim of the first experiment was to determine whether the use of the consensus-based recommendation generation algorithm could enable a recommending registry to generate effective recommendations under normal, non-spamming conditions. Of particular interest was which CB variations caused the registry to be most effective, and how this effectiveness compared against that of SCO.

Setup and Presentation

Setup The prototype registry was assessed in terms of 8 sub-experiments. For each sub-experiment, the registry was configured to use a different situation attribute subset of HourOfDay, Location and Role. It was then configured to use, and evaluated for, 28 different recommendation generation algorithms: SCO and 27 CB variations. In terms of the CB variations, the registry was configured to use each of the 9 rank aggregation methods (algorithm step 2), with either no local Kemenisation, LK or InfLK applied (algorithm step 3). Since it can safely be assumed that no spamming occurs in the DCS printer scenario, the evaluation results demonstrate how the CB variations perform under normal conditions, against each other and SCO.

Presentation For each sub-experiment, the NCIOC values (at tolerance level 5) of all 28 differing registry configurations are presented in a table. The tables for 6 of the sub-experiments are given in Figures 10.6 to 10.11; the tables for No Attributes and HourOfDay are given in Appendix C, in Figures C.1 and C.2 respectively. The table results of each sub-experiment are presented in 10 rows. The top row refers to the single SCO configuration, whilst each row below refers to the three CB configurations that use a particular rank aggregation method; "Basic" is the method with no local Kemenisation applied, "+LK" is the method with LK applied, and "+InfLK" the method with InfLK applied. For example, the table cell [MC2, +InfLK] contains the NCIOC value of the prototype registry when configured to use the CB variation of the MC_2 rank aggregation method with InfLK applied

(referred to as InfLK-MC_2). For each rank aggregation method, the cells of the three corresponding configurations (Basic, LK and InfLK) are colour coded in terms of relative effectiveness. In order of least to most effectiveness (i.e. smallest to largest NCIOC value), the three configurations are coloured light grey, medium grey and dark grey. The NCIOC value of the most effective configuration is also given in the "Best NCIOC" cell. The "Rank" cell gives the rank of the rank aggregation method if the different methods (and SCO) are ordered by "Best NCIOC" value, most to least effective. Thus, the Rank column provides a rough indication of how the different rank aggregation methods perform against each other and SCO. As a clarifying example, consider the MC_2 rank aggregation method in the Figure 10.6 (Location) table. InfLK-MC_2 (dark grey) is more effective than Basic-MC_2 (medium grey), which is more effective than LK-MC_2 (light grey). Consequently, the "Best NCIOC" of MC_2 is that of InfLK-MC_2 (0.7583), and MC_2 is ranked 7th out of the rank aggregation methods and SCO in terms of effectiveness.

Results

A number of general trends concerning the CB variations can be identified from the 6 sub-experiment tables shown in Figures 10.6 to 10.11. It should be noted that these trends are not reflected to such an extent in the other 2 sub-experiment tables given in the appendix. These 2 other tables refer to the prototype registry when it was configured to use the situation attribute subsets of HourOfDay or No Attributes. As can be seen from the results of the last chapter (see Figures 9.4 and 9.5), the prototype registry is consistently much less effective when configured to use either subset in assessing service selection situation-similarity. Either subset presumably leads to irrelevant service selections being identified, which are predominately inappropriate courses of action for a requesting consumer. Based on such bad data, these is a limit to how effective generated recommendations can ever be, regardless of the particular recommendation generation algorithm used. In the case of the consensus-based recommendation generation algorithm, such irrelevant service selections are presumably too disparate to allow a meaningful consensus opinion of service appropriateness to be formed. For this chapter, therefore, the 2 sub-experiments associated with HourOfDay and No Attributes are ignored. In reality, a deployed recommending registry would not be configured to use such ineffective situation attribute subsets.

Recommendation Generation Algorithm	NCIOC (to tolerance level 5)			Best NCIOC	Rank
	Basic	+LK	+InfLK		
SCO	0.7605	N/A	N/A	0.7605	1
Borda	0.7596	0.7449	0.7588	0.7596	2
MC1	0.7313	0.7094	0.7584	0.7584	6
InfMC1	0.7592	0.7397	0.7594	0.7594	3
MC2	0.7337	0.7093	0.7583	0.7583	7
InfMC2	0.759	0.7398	0.7589	0.759	4
MC3	0.7354	0.709	0.758	0.758	8
InfMC3	0.7559	0.744	0.7585	0.7585	5
MC4	0.6915	0.6934	0.7575	0.7575	10
InfMC4	0.7454	0.7377	0.7575	0.7575	9

Figure 10.6: Experiment One - Location

Recommendation Generation Algorithm	NCIOC (to tolerance level 5)			Best NCIOC	Rank
	Basic	+LK	+InfLK		
SCO	0.7234	N/A	N/A	0.7234	9
Borda	0.7235	0.6863	0.7239	0.7239	3
MC1	0.6965	0.6899	0.7241	0.7241	1
InfMC1	0.7238	0.684	0.7237	0.7238	5
MC2	0.701	0.6906	0.724	0.724	2
InfMC2	0.7237	0.6838	0.7237	0.7237	7
MC3	0.7007	0.6902	0.7239	0.7239	4
InfMC3	0.7222	0.691	0.7224	0.7224	10
MC4	0.6427	0.6538	0.7238	0.7238	6
InfMC4	0.6957	0.686	0.7236	0.7236	8

Figure 10.7: Experiment One - HourOfDay & Location

Recommendation Generation Algorithm	NCIOC (to tolerance level 5)			Best NCIOC	Rank
	Basic	+LK	+InfLK		
SCO	0.6867	N/A	N/A	0.6867	2
Borda	0.6863	0.6781	0.6848	0.6863	4
MC1	0.6779	0.6769	0.6842	0.6842	6
InfMC1	0.6861	0.6775	0.6869	0.6869	1
MC2	0.6779	0.6769	0.684	0.684	7
InfMC2	0.6847	0.6771	0.6855	0.6855	5
MC3	0.6776	0.6764	0.683	0.683	10
InfMC3	0.6855	0.6774	0.6866	0.6866	3
MC4	0.6723	0.6739	0.6837	0.6837	8
InfMC4	0.6806	0.6767	0.6835	0.6835	9

Figure 10.8: Experiment One - Location & Role

Recommendation Generation Algorithm	NCIOC (to tolerance level 5)			Best NCIOC	Rank
	Basic	+LK	+InfLK		
SCO	0.6598	N/A	N/A	0.6598	1
Borda	0.6296	0.6128	0.6385	0.6385	9
MC1	0.4818	0.4541	0.6414	0.6414	3
InfMC1	0.6404	0.5777	0.6399	0.6404	7
MC2	0.5045	0.4636	0.6411	0.6411	6
InfMC2	0.6331	0.5668	0.6432	0.6432	2
MC3	0.52	0.4767	0.6411	0.6411	5
InfMC3	0.6303	0.6166	0.6382	0.6382	10
MC4	0.474	0.4711	0.6412	0.6412	4
InfMC4	0.605	0.5571	0.6397	0.6397	8

Figure 10.9: Experiment One - Role

Recommendation Generation Algorithm	NCIOC (to tolerance level 5)			Best NCIOC	Rank
	Basic	+LK	+InfLK		
SCO	0.6395	N/A	N/A	0.6395	5
Borda	0.6396	0.6025	0.6398	0.6398	4
MC1	0.5151	0.5009	0.6357	0.6357	6
InfMC1	0.64	0.5974	0.6365	0.64	3
MC2	0.5298	0.5109	0.6356	0.6356	8
InfMC2	0.6425	0.5971	0.6368	0.6425	1
MC3	0.5315	0.5115	0.6357	0.6357	7
InfMC3	0.6413	0.6054	0.6362	0.6413	2
MC4	0.4432	0.4487	0.6356	0.6356	8
InfMC4	0.5758	0.5506	0.6353	0.6353	9

Figure 10.10: Experiment One - HourOfDay & Role

Recommendation Generation Algorithm	NCIOC (to tolerance level 5)			Best NCIOC	Rank
	Basic	+LK	+InfLK		
SCO	0.6317	N/A	N/A	0.6317	1
Borda	0.6312	0.5997	0.631	0.6312	5
MC1	0.6189	0.6098	0.6309	0.6309	6
InfMC1	0.6314	0.5994	0.6316	0.6316	2
MC2	0.6221	0.6098	0.6308	0.6308	7
InfMC2	0.6309	0.5993	0.6313	0.6313	4
MC3	0.6221	0.6097	0.6308	0.6308	7
InfMC3	0.6311	0.6014	0.6315	0.6315	3
MC4	0.5708	0.5782	0.6308	0.6308	7
InfMC4	0.6099	0.6012	0.6308	0.6308	7

Figure 10.11: Experiment One - HourOfDay & Location & Role

Comparing Rank Aggregation Methods The first set of general trends relate to the rank aggregation methods used in the CB variations, and can be identified from the "Basic" column of each sub-experiment table. The consensus-based recommendation generation algorithm actually requires that some form of local Kemenisation be applied (in step 3) to the aggregation calculated by a rank aggregation method (in step 2). However, it seemed worthwhile to consider the "raw" performance of the rank aggregation methods against each another. The first trend is that the non-Inf version of every MC-based rank aggregation method (e.g. MC_1) is consistently less effective than the Inf version (e.g. InfMC_1). Thus, it appears that, under normal conditions, it is better to calculate an aggregation (and thus a recommendation) based on complete, and possibly inferred, voter preference rankings than on incomplete voter preference rankings. Presumably, this degradation in performance between the Inf and the non-Inf version of every MC-based rank aggregation method can be attributed to the undesirable qualities of the non-Inf version that were discussed earlier in Section 10.4.3. In other words, the Inf versions are presumably more able to mitigate the impact of any voters with idiosyncratic, generally inappropriate, behaviour. The second associated trend is that MC_4 is consistently the least effective of all 9 rank aggregation methods.

Comparing Versions of Local Kemenisation The second set of general trends relate to the 2 versions of local Kemenisation used in the CB variations, LK and InfLK, and can be identified from the "+LK" and "+InfLK" columns of each sub-experiment table. The first trend is that, for a particular rank aggregation method, the CB variation that uses InfLK (e.g. InfLK-Borda) is more effective than the one that uses LK (e.g. LK-Borda). Thus, it appears that, under normal conditions, it is better to locally Kemenise the aggregation calculated by the rank aggregation method based on complete, possibly inferred, voter preference rankings than on incomplete voter preference rankings. Presumably, this difference in performance between LK and InfLK can be attributed to the undesirable qualities of LK that were discussed earlier in Section 10.4.4. In other words InfLK is less affected by voters with idiosyncratic, generally inappropriate behaviour. The second associated trend is that, of the 3 CB variations (Basic, LK, InfLK) that use a particular rank aggregation method, the one which uses LK (e.g. LK-Borda) is predominately the least effective, less than if no local Kemenisation is applied (e.g. Basic-Borda)!

Considering InfLK Given that the prototype registry was more effective when configured to use InfLK rather LK, it was decided to concentrate attention on general trends relating to InfLK alone. These trends can be identified from the "+InfLK" column of each sub-experiment table. The primary trend is that, for a particular situation attribute subset (sub-experiment), there is very little difference between the InfLK-using CB variations in terms of effectiveness, despite the different rank aggregation methods used. For example, for Location (Figure 10.6), the difference between the NCIOC values of the most effective InfLK-using CB variation (InfLK-InfMC_1: 0.7594) and the least effective (InfLK-MC_4: 0.7575) is only 0.0019. For all the 6 considered sub-experiments, the smallest difference is 0.0008 (HourOfDay & Location & Role: Figure 10.11), and largest is 0.0050 (Role: Figure 10.10). This is interesting, given that there is much greater difference between CB variations (with their different rank aggregation methods) when no local Kemenisation is used. Returning to Location, the difference between the NCIOC values of the most effective CB variation using no local Kemenisation (Basic-Borda: 0.7596) and the least effective (Basic-MC_4: 0.6915) is 0.0681. Presumably, given that a similar trend does not occur for LK-using CB variations, the trend can be ascribed to the fact that InfLK is based on complete, possibly inferred, voter preference rankings. By implication, if InfLK is used in step 3 of the consensus-based recommendation generation algorithm, the actual rank aggregation method used in step 2 would seem to be of lesser importance.

Comparing CB and SCO In all considered sub-experiments, the InfLK-using CB variations compare very favourably against SCO in terms of effectiveness. For example, for Location (Figure 10.6), the NCIOC value of SCO is 0.7605 whilst that of the best InfLK-using CB variation (InfLK-InfMC_1) is 0.7594, a difference of only 0.0011. Even better, for HourOfDay & Location (Figure 10.7), the NCIOC value of the best InfLK-using CB variation (InfLK-MC_1), at 0.7241, is slightly higher than that of SCO, at 0.7234, an increase of 0.0007. In fact, for 3 out of the 6 considered sub-experiments, an InfLK-using CB variation performs better than SCO.

Conclusion

It has been demonstrated that the consensus-based recommendation generation algorithm, using InfLK, does indeed enable a recommending registry to be as effective under normal non-spamming conditions as when SCO is used.

10.5.2 Experiment Two - CB Under Spamming Conditions

The aim of the second experiment was to determine whether, compared to SCO, the use of the consensus-based recommendation generation algorithm could enable a recommending registry to remain effective to a greater extent in the face of spamming.

Setup and Presentation

General Setup Different configurations of the prototype recommending registry were each evaluated under comparable non-spamming and simulated spamming conditions, and the corresponding degradation in effectiveness assessed. As has been previously noted, spamming should not normally occur in the DCS printer scenario, but it is possible to simulate recommending registry spamming in an authentic manner; precise details are given below. Like the first experiment, the prototype registry was assessed in terms of 8 sub-experiments. For each sub-experiment, the registry was configured to use a different situation attribute subset of HourOfDay, Location and Role. It was then configured to use, and evaluated for, 4 different recommendation generation algorithms: SCO, and the 3 CB variations of Basic-InfMC$_1$ (InfMC$_1$ with no local Kemenisation applied), LK-InfMC$_1$ and InfLK-InfMC$_1$. Essentially, to ensure that this experiment remained tractable, these 3 CB variations were chosen to be representative of the 27 CB variations assessed in Experiment One. The 2 types of local Kemenisation and the "no local Kemenisation" option are represented, whilst InfMC$_1$ seemed to be (marginally) the most effective of the 9 rank aggregation methods in the Experiment One, given that it had the best average rank (from the "Rank" column) across the 8 sub-experiments.

Setup for Non-spamming and Simulated Spamming Conditions For a particular sub-experiment in this experiment, the 4 different registry configurations were each evaluated under comparable non-spamming and simulated spamming conditions in the following manner. Firstly, in preparation, the prototype registry was configured to use SCO, and was then used to generate recommendations for each of the 10397 ESSEs in the standard DCS printer scenario ESSE set. This ESSE set was then "trimmed" to contain only those ESSEs for which "proper" recommendations were successfully generated, which were based on a non-zero number of identified relevant service selections. In other words, those ESSEs for which no relevant service selections were identified, and for which a recommendation consequently consisted of all the available type-matching services in a single

tied rank, were filtered out. This trimmed ESSE set will be referred to as the "successful" ESSE set. Next, for each ESSE in this set, the corresponding SCO-generated recommendation was assessed, and the 10 worst-ranked services were identified; these services will be referred to as the "bottom" services of an ESSE.

Then, to assess a registry configuration under non-spamming conditions, it was evaluated as normal for the recommendations it generated in response to the ESSEs in the successful ESSE set. To assess the registry configuration under comparable spamming conditions, it was also evaluated for the recommendations it generated in response to the same ESSEs. However, for each ESSE, the relevant service selections identified during the recommendation generation process were intercepted, and augmented with a set of simulated sham service selections deliberately designed to boost the recommendation ranks of the corresponding 10 bottom services. The recommendation generated for the ESSE was then based on this augmented set of relevant service selections. More precisely, the added sham selections were simulated as being made by a single spamming individual, and would cause the bottom services to be boosted into the top 10 ranks of a SCO-generated recommendation, in a particular order. The particular sequence in which the sham selections were made would also cause the bottom services to be boosted into the top 10 ranks (in the same order) of the spamming individual's preference ranking calculated in a CB variation using RPVoter (with offsets of 5). Thus, through this process of augmentation, it was possible to simulate recommending registry spamming.

In overall terms, therefore, the prototype registry was simulated as being spammed in every situation in which an ESSE occurred. For a particular situation, the registry was being spammed for 10 particular services by a single individual. Note that the spam services may have differed between situations. Thus, since each registry configuration, with its different recommendation generation algorithm, was evaluated for the same set of ESSEs under the same non-spamming and spamming conditions, it was possible to ascertain from the evaluation results the relative robustness of the algorithms in the face of spamming.

Presentation For each sub-experiment, the evaluation results for the 4 registry configurations under both non-spamming and simulated spamming conditions are presented in the standard graph/table format used in the last chapter, which was explained in Section 9.2. The results for the Location and HourOfDay & Location sub-experiments are given in

Figures 10.12 and 10.13 respectively; the results for the other 6 sub-experiments are given in Appendix C, in Figures C.3 to C.8. For a particular sub-experiment, the evaluation details of a registry configuration under non-spamming conditions are referred to by the name of the particular recommendation generation algorithm used (e.g. "LK-InfMC$_1$"); under spamming conditions they are referred to by this name followed by "(spammed)" (e.g. "LK-InfMC$_1$ (spammed)"). It should be noted that InfMC$_1$ with no local Kemenisation applied is simply referred to as "InfMC$_1$", rather than "Basic-InfMC$_1$".

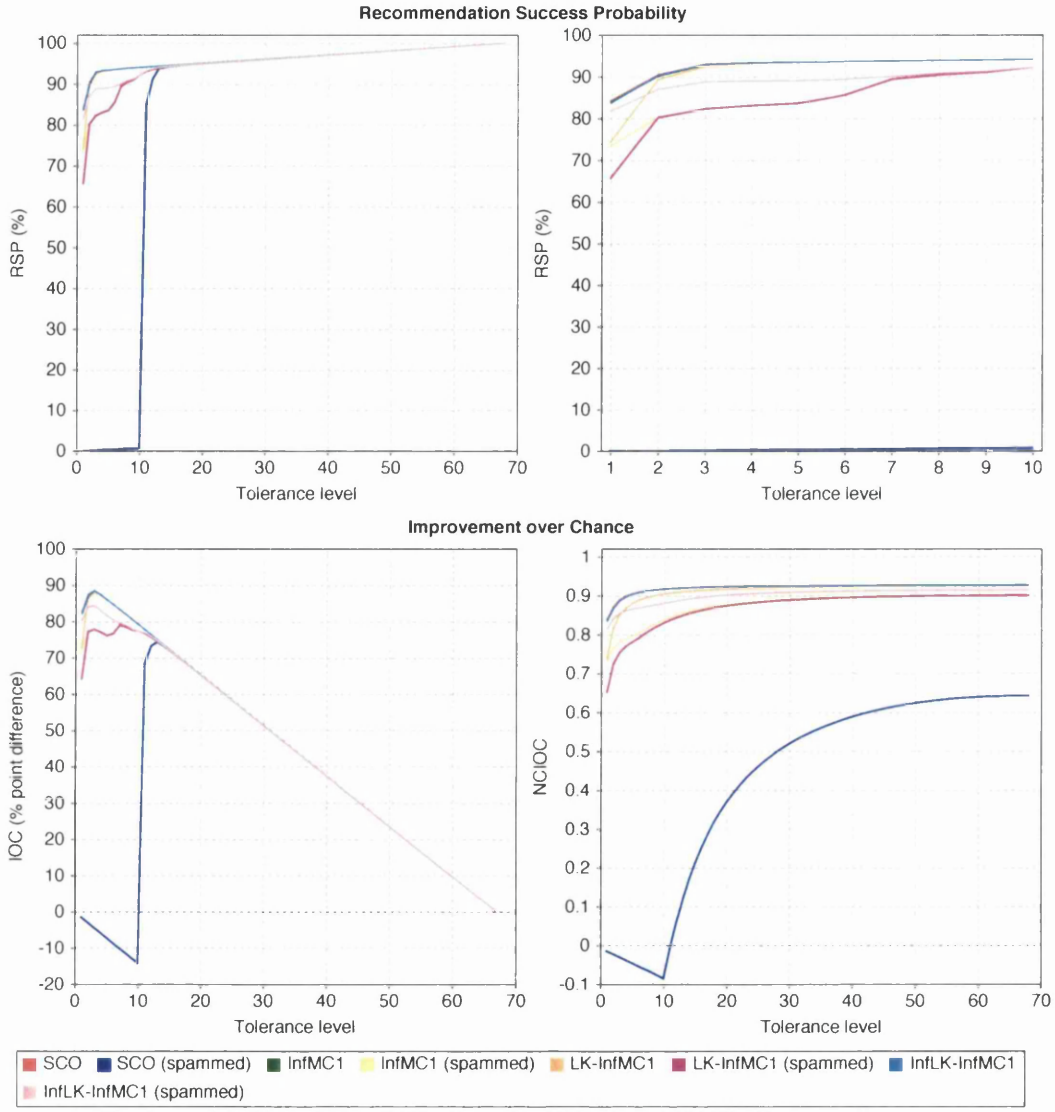
Results

A number of general trends concerning the recommendation generation algorithms can be identified from the 6 sub-experiments considered; as before, the 2 sub-experiments associated with HourOfDay and No Attributes are ignored.

Considering SCO Firstly, as expected, the use of SCO causes registry effectiveness to degrade massively in the face of spamming. In the case of Location (Figure 10.12), for example, the NCIOC value of the SCO configuration drops from 0.905 under non-spamming conditions to -0.0447 under spamming conditions, a difference of 0.9497! Indeed, for all 6 considered sub-experiments, the NCIOC value of SCO is negative under spamming conditions. This trend corroborates the assertion made earlier as to the inadequacy of SCO in the face of spamming.

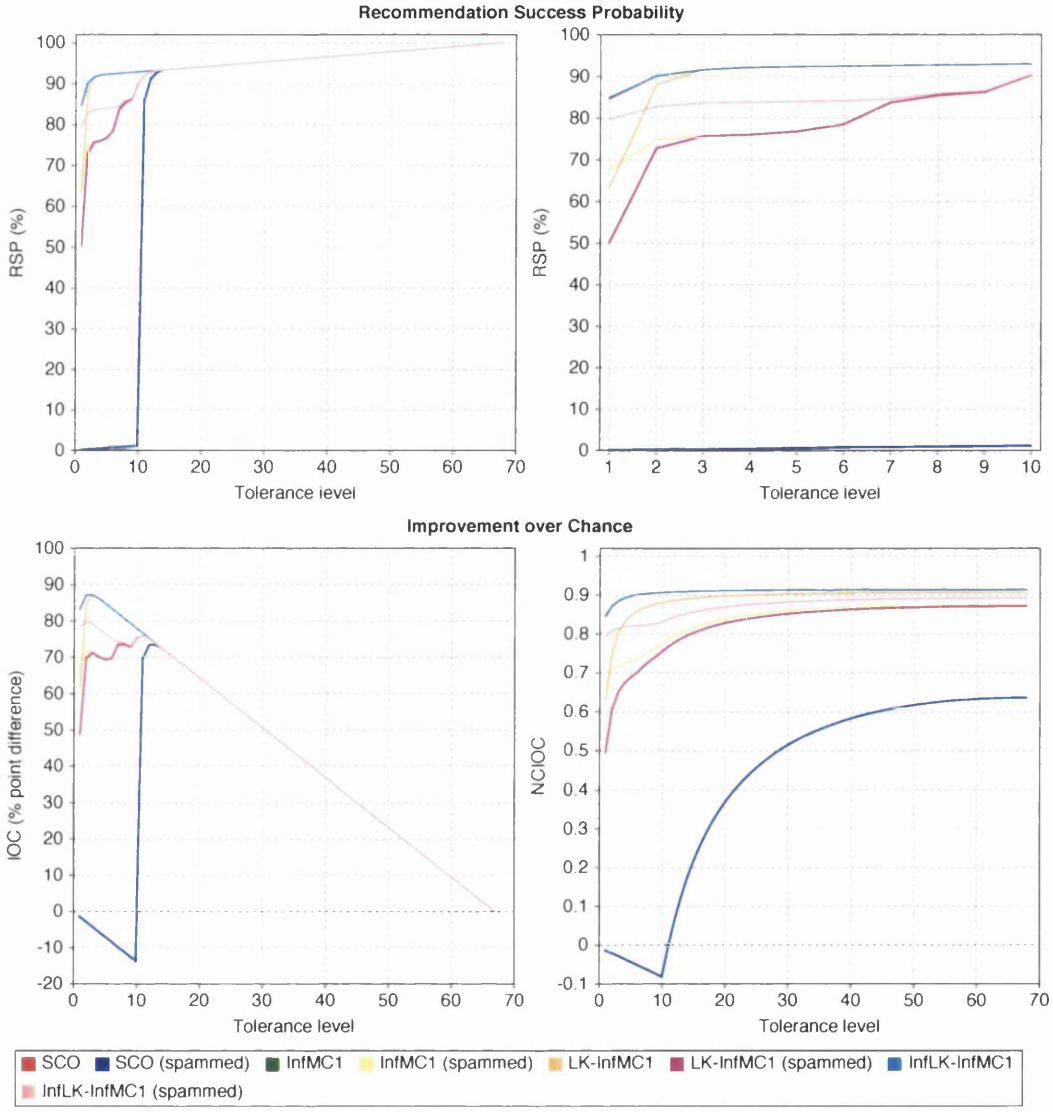
Comparing CB and SCO Secondly, compared to SCO, the use of any CB variation causes registry effectiveness to degrade to a much lesser extent in the face of spamming. Again, for Location, the largest drop in NCIOC value for a CB variation (InfMC$_1$) between non-spamming (0.9034) and spamming conditions (0.7963) is 0.1071, much smaller than the corresponding drop for SCO (0.9497). Across the 6 considered sub-experiments, the smallest NCIOC value of a CB variation under spamming conditions is 0.5148 (LK-InfMC$_1$ for HourOfDay & Role; see Figure C.5), some way above the negative values of SCO. Thus, the consensus-based recommendation generation algorithm is indeed significantly more robust than SCO in the face of spamming.

Considering Local Kemenisation Thirdly, for the CB variations, local Kemenisation of either form causes registry effectiveness to degrade to a lesser extent than if it were not applied, with the smallest degradation occurring when InfLK rather than LK is used.



Recommendation Generation Algorithm	Recommendations			Successful Recommendations Info			Cumul. IOC (level 5)	NCIOC (level 5)	Rank
	Success	Failure		Mean # Users	Mean # Services Selected	Mean # Service Selections			
		No service selections	No user situation						
SCO	8737	0	0	63.56	5.29	3449.91	432.24	0.905	1
SCO (spammed)	8737	0	0	64.56	15.29	35714.24	-21.35	-0.0447	8
InfMC1	8737	0	0	63.56	5.29	3449.91	431.48	0.9034	3
InfMC1 (spammed)	8737	0	0	64.56	15.29	35714.24	380.34	0.7963	6
LK-InfMC1	8737	0	0	63.56	5.29	3449.91	420.39	0.8802	4
LK-InfMC1 (spammed)	8737	0	0	64.56	15.29	35714.24	372.8	0.7805	7
InfLK-InfMC1	8737	0	0	63.56	5.29	3449.91	431.62	0.9037	2
InfLK-InfMC1 (spammed)	8737	0	0	64.56	15.29	35714.24	413.49	0.8657	5

Figure 10.12: Experiment Two - Location



Recommendation Generation Algorithm	Recommendations			Successful Recommendations Info			Cumul. IOC (level 5)	NCIOC (level 5)	Rank
	Success	Failure		Mean # Users	Mean # Services Selected	Mean # Service Selections			
		No service selections	No user situation						
SCO	8380	0	0	42.7	2.83	324.84	428.67	0.8975	3
SCO (spammed)	8380	0	0	43.7	12.83	3429.39	-20.58	-0.0431	8
InfMC1	8380	0	0	42.7	2.83	324.84	428.9	0.898	1
InfMC1 (spammed)	8380	0	0	43.7	12.83	3429.39	348.79	0.7303	6
LK-InfMC1	8380	0	0	42.7	2.83	324.84	405.33	0.8486	4
LK-InfMC1 (spammed)	8380	0	0	43.7	12.83	3429.39	329.21	0.6893	7
InfLK-InfMC1	8380	0	0	42.7	2.83	324.84	428.87	0.8979	2
InfLK-InfMC1 (spammed)	8380	0	0	43.7	12.83	3429.39	391.82	0.8204	5

Figure 10.13: Experiment Two - HourOfDay & Location

Again, for Location, the corresponding drops in NCIOC value between non-spamming and spamming conditions for $InfMC_1$ (with no local Kemenisation applied), LK-$InfMC_1$ and InfLK-$InfMC_1$ are 0.1071, 0.0997 and 0.038 respectively. Indeed, across the 6 considered sub-experiments, the largest drops for $InfMC_1$, LK-$InfMC_1$ and InfLK-$InfMC_1$ are 0.2145, 0.2116 and 0.1296 respectively (all for HourOfDay & Location & Role; see Figure C.6). Thus, the anti-spamming benefits of local Kemenisation claimed by Dwork et al are corroborated. The better performance of InfLK compared to LK can presumably be attributed to the undesirable qualities of LK discussed earlier in Section 10.4.4. The InfLK-$InfMC_1$-using registry configuration actually has the highest NCIOC value under spamming conditions for all 6 considered sub-experiments.

Conclusion

It has been demonstrated that, compared to SCO, the consensus-based recommendation algorithm does indeed enable a recommendation registry to remain effective to a greater extent in the face of spamming. Moreover, the algorithm has been demonstrated to be most robust when InfLK is used.

10.5.3 Experiment Three - CB Under Collective Spamming Conditions

The aim of the third experiment was to investigate how the consensus-based recommendation generation algorithm performed in the face of more orchestrated spamming. In Experiment Two, the simulated spamming conditions used to evaluate the prototype registry configurations took the form of a *single* individual spamming the registry for certain services in a situation. Thus, it was decided to investigate how the CB algorithm performed if *multiple* individuals collectively spammed the registry for the same services in a situation.

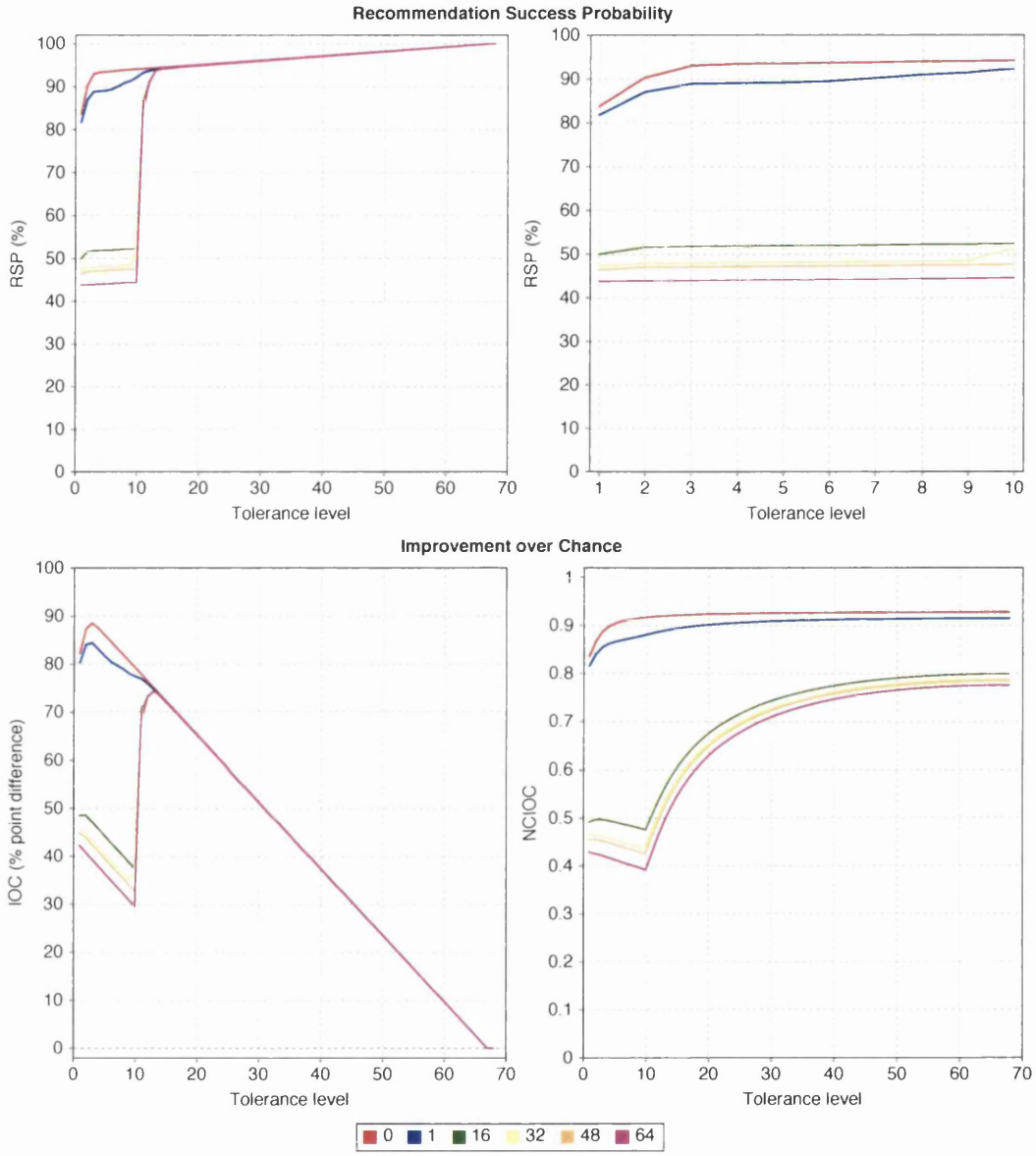
Setup and Presentation

Setup The experiment takes a very similar form to Experiment Two. The prototype registry was assessed in terms of 8 sub-experiments. For each sub-experiment, the registry was configured to use a different situation attribute subset of HourOfDay, Location and Role, and the CB variation of InfLK-$InfMC_1$. This CB variation was chosen as representative of all those investigated previously, since it had proved most robust in Experiment Two. The single registry configuration was then evaluated under various simulated spamming condi-

tions, in the same manner as in Experiment Two. After preparation, the configuration was evaluated in terms of the recommendations it generated in response to the ESSEs of the "trimmed" successful ESSE set. As before, for each ESSE, the relevant service selections identified during the recommendation process were intercepted, and augmented with a set of simulated sham service selections deliberately designed to boost the recommendation ranks of the corresponding 10 bottom services. The recommendation generated for the ESSE was then based on this augmented set of relevant service selections. However, in this experiment, the added sham selections were simulated as being made by *multiple* individuals. Essentially, the same sequence of sham services was made as in Experiment Two, but was "cloned" for multiple individuals. So, for n individuals, a sham selection was made n times, once by each individual. The particular values of n used will be discussed below.

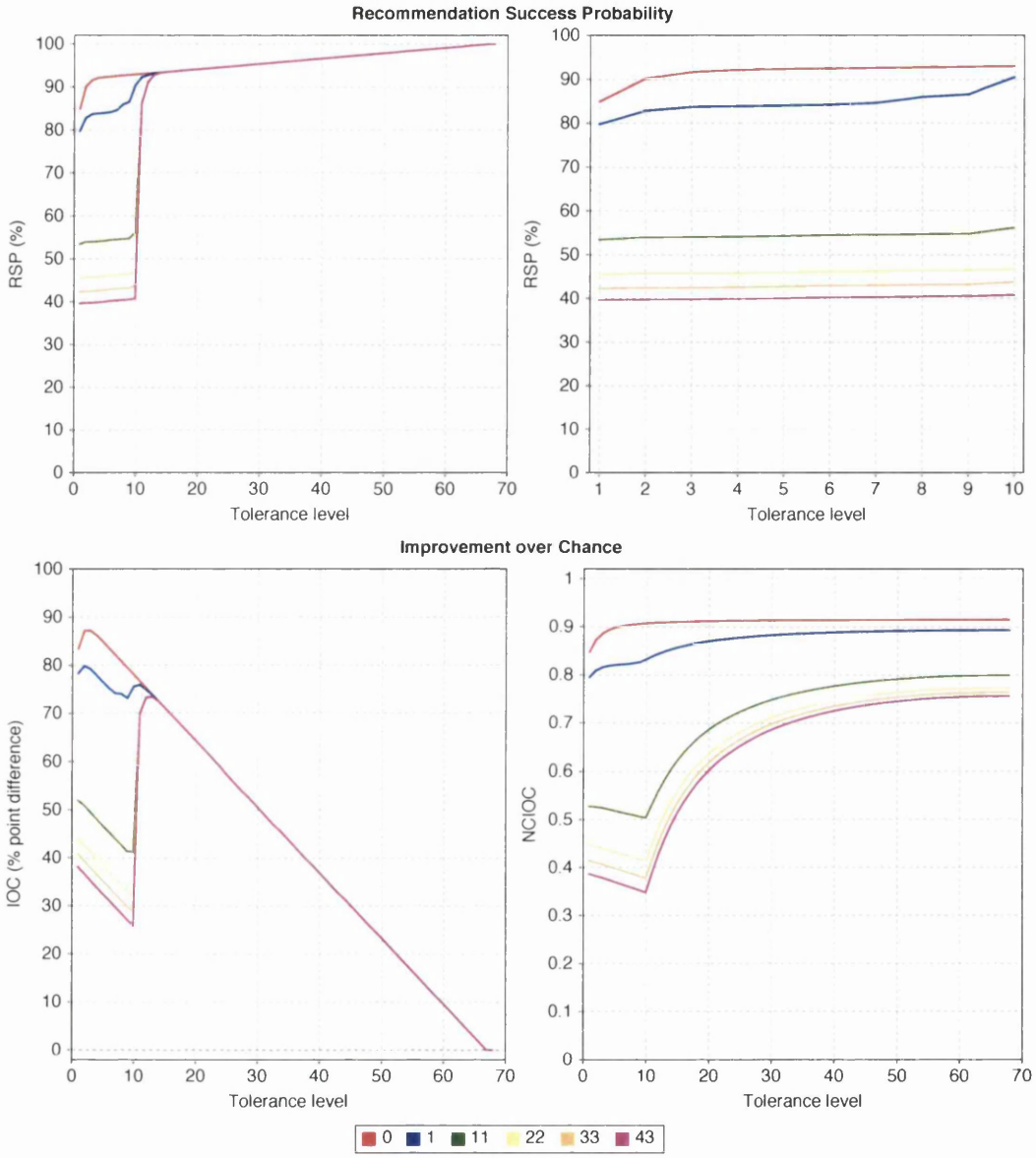
In overall terms, therefore, the prototype registry was simulated as being collectively spammed in every situation in which an ESSE occurred. For a particular situation, the registry was spammed for 10 particular services by n individuals. Note that the spam services may have differed between situations. The registry configuration was evaluated in this way for 6 different values of n. The first 2 values were 0 (no spamming) and 1. The other 4 values were calculated from the average number of selecting individuals (voters) responsible for the relevant service selections on which an ESSE recommendation was based (under normal non-spamming conditions, i.e. $n = 0$). If this average number of voters is v, then n was set to $\frac{v}{4}, \frac{v}{2}, \frac{3v}{4}$ and v, each rounded up to a whole number. Thus, the registry configuration was being evaluated for spamming conditions where an increasing proportion of the voters, whose inferred opinions (preference rankings) were being aggregated by InfLK-InfMC$_1$ to form a recommendation, were collectively spamming the registry.

Presentation For each sub-experiment, the evaluation results for the single registry configuration under the 6 different spamming conditions are presented in the same format as in Experiment Two. The results for the Location and HourOfDay & Location sub-experiments are given in Figures 10.14 and 10.15 respectively; owing to its lack of relevance, the No Attributes sub-experiment was not performed, but the results for the other 5 sub-experiments are given in Appendix C, in Figures C.9 to C.13.



# Spamming Voters	Recommendations			Successful Recommendations Info			Cumul. IOC (level 5)	NCIOC (level 5)	Rank
	Success	Failure		Mean # Users	Mean # Services Selected	Mean # Service Selections			
		No service selections	No user situation						
0	8737	0	0	63.56	5.29	3449.91	431.62	0.9037	1
1	8737	0	0	64.56	15.29	35714.24	413.49	0.8657	2
16	8737	0	0	79.56	15.29	519679.26	234.63	0.4912	3
32	8737	0	0	95.56	15.29	1035908.61	216.89	0.4541	4
48	8737	0	0	111.56	15.29	1552137.97	212.31	0.4445	5
64	8737	0	0	127.56	15.29	2068367.32	197.18	0.4128	6

Figure 10.14: Experiment Three - Location



# Spamming Voters	Recommendations			Successful Recommendations Info			Cumul. IOC (level 5)	NCIOC (level 5)	Rank
	Success	Failure		Mean # Users	Mean # Services Selected	Mean # Service Selections			
		No service selections	No user situation						
0	8380	0	0	42.7	2.83	324.84	428.87	0.8979	1
1	8380	0	0	43.7	12.83	3429.39	391.82	0.8204	2
11	8380	0	0	53.7	12.83	34474.84	247.44	0.5181	3
22	8380	0	0	64.7	12.83	68624.84	206.43	0.4322	4
33	8380	0	0	75.7	12.83	102774.84	190.26	0.3983	5
43	8380	0	0	85.7	12.83	133820.29	176.65	0.3699	6

Figure 10.15: Experiment Three - HourOfDay & Location

Results

A key trend can be identified from the 6 sub-experiments considered; as before, the sub-experiment associated with HourOfDay is ignored. Unsurprisingly, the trend is that the more selecting individuals (voters) that collectively spam the registry, the more registry effectiveness is diminished. For example, in the case of Location (Figure 10.14), the drop in NCIOC value between non-spamming (0.9037) and spamming conditions is only 0.038 when $n = 1$ (NCIOC = 0.8657), but is 0.4496 when $n = 32$ (NCIOC = 0.4541). Thus, the consensus-based recommendation generation algorithm performs less effectively when more voters collectively spam a registry. This is to be expected, since the larger the proportion of collectively spamming voters, the less their opinions can be effectively countered by those of the remaining proportion of non-spamming voters.

However, it should be noted that such orchestrated spamming would probably be an unlikely occurrence. It may be relatively straightforward to construct such spamming conditions in simulation, but in real life numerous people would need to organise themselves into making similar sequences of sham service selections within the same situation in the same time-frame. If they wished to continue spamming the recommending registry continuously, this would require even more organisation over a much longer period of time. The other positive aspect that can be derived from the 6 considered sub-experiments in this experiment is that, even for the largest numbers of n collectively spamming individuals, the CB algorithm (i.e. InfLK-InfMC$_1$) performed much better than SCO did when it was faced with only one spamming individual in Experiment Two (where $n = 1$). The lowest NCIOC value of the CB-using registry across the 6 sub-experiments is 0.3185 ($n = 52$ for HourOfDay & Role; see Figure C.11); as was noted earlier, for all 6 considered sub-experiments in Experiment Two, the NCIOC values of the SCO-using registry were all negative.

10.6 Conclusion

In conclusion, it can be seen that the consensus-based recommendation generation algorithm (CB) is a more suitable choice for the CF-based recommending registry design than SCO. Through the experiments performed, it has been demonstrated that, compared to SCO, the more robust CB algorithm can enable a recommending registry not only to be effective under normal conditions, but also to remain effective to a greater extent in the

face of spamming. Moreover, CB has been demonstrated to perform best under both non-spamming and spamming conditions when InfLK is used in algorithm step 3. When InfLK is used, it would appear that the actual rank aggregation method used in step 2 is of lesser importance. However, at least when evaluated under non-spamming conditions, $InfMC_1$ did perform marginally better than the other rank aggregation methods. Thus, InfLK-$InfMC_1$ could be considered the ideal CB variation.

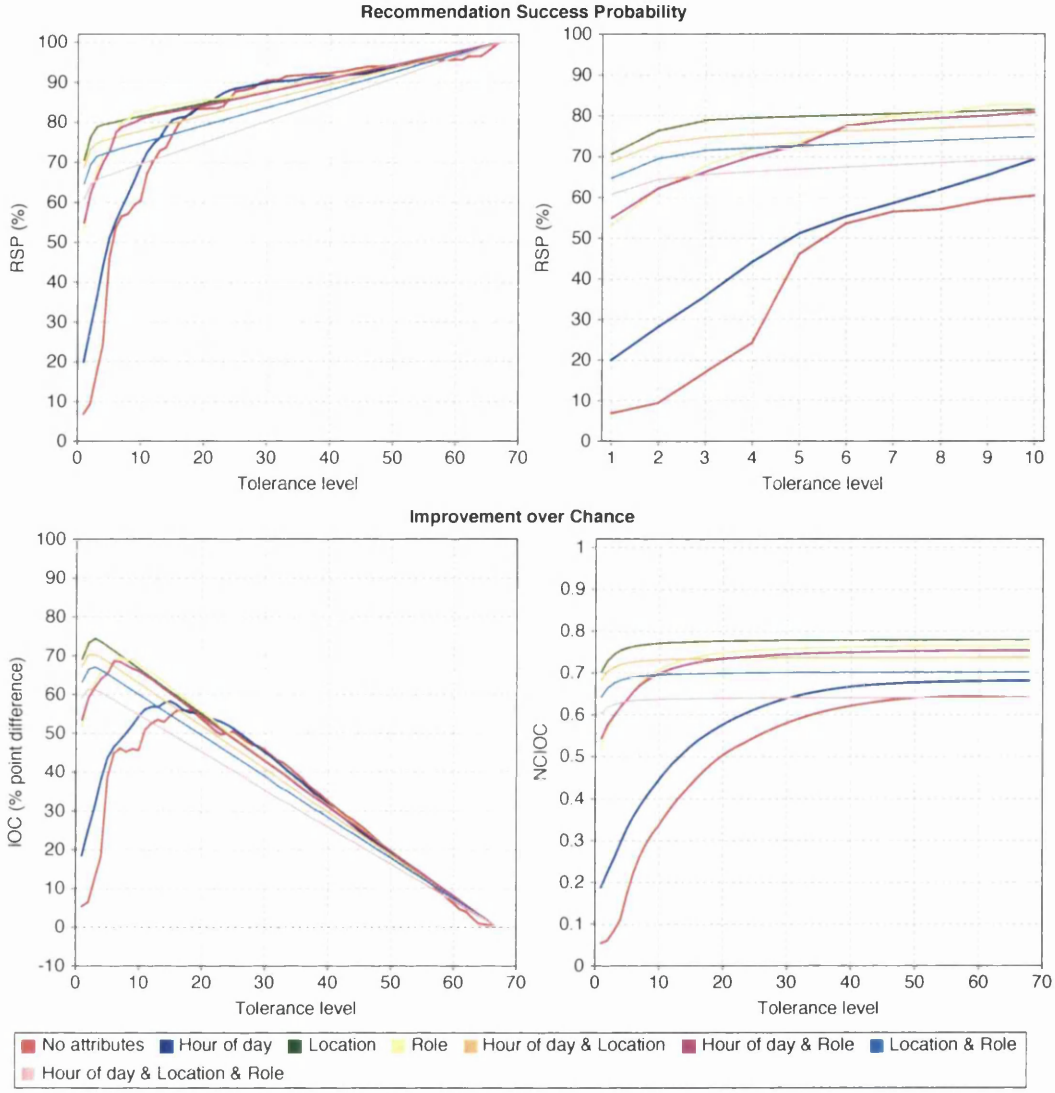
NB: For clarification purposes, the evaluation results obtained in Experiment One (see Section 10.5.1) for the 8 configurations of the prototype registry which used InfLK-$InfMC_1$ are shown in Figure 10.16. The configurations differ in the printer-specific situation attribute subset used.

10.7 Summary

In this chapter, attention has been focused on my investigation and development of an advanced recommendation generation algorithm to replace SCO. The need for a more robust algorithm was motivated at the beginning of the chapter through an explanation of how the effectiveness of a design-adhering recommending registry that used SCO could be diminished as a result of spamming.

Following a description and justification of my proposal for a style of algorithm based on consensus, mention was made of the fact that I had needed to widen my investigation beyond the areas of service discovery and CF in order to find research of relevance to such a concept. It was noted that I had identified two research areas, Social Choice Theory (Economics) and meta-search (Computing Science), where some research, in the form of rank aggregation methods, was relevant.

The bulk of the chapter has been concerned with the definition and detailed explanation of the consensus-based recommendation generation algorithm (CB) that I developed. Details of the two original methods (algorithm step 1: SCOVoter and RPVoter) I devised to calculate voter preference rankings have been given, followed by details of the five rank aggregation methods (algorithm step 2: Borda and MC_1 - MC_4) adapted by me for use in CB. The concept of local Kemenisation has been explained and my adaptation of it for use in CB (algorithm step 3) has been defined. Finally, details have been given of the three experiments that were performed to evaluate variations of CB under both normal and spamming conditions. The conclusion has ultimately been drawn that CB is a more



Situation Attribute Subset	Recommendations			Successful Recommendations Info			Cumul. IOC (level 5)	NCIOC (level 5)	Rank
	Success	Failure		Mean # Users	Mean # Services Selected	Mean # Service Selections			
		No service selections	No user situation						
No attributes	10397	0	0	868.54	64.96	39342.84	81.18	0.17	8
Hour of day	10397	0	0	460.67	45.83	3257.31	156.84	0.3284	7
Location	8737	1419	241	63.56	5.29	3449.91	362.71	0.7594	1
Role	10360	0	37	75	13.96	4865.22	305.61	0.6399	4
Hour of day & Location	8380	1776	241	42.7	2.83	324.84	345.67	0.7237	2
Hour of day & Role	10268	92	37	51.31	8.32	422.77	303.98	0.6365	5
Location & Role	7824	2332	241	64.32	4.1	3522.77	328.1	0.6869	3
Hour of day & Location & Role	7288	2868	241	46.07	2.62	349.51	301.68	0.6316	6

Figure 10.16: Experiment One - InfLK-InfMC$_1$

suitable choice for the CF-based recommending registry design than SCO.

Following this first successful development of the basic recommending registry design, a second improvement, that of relaxing the test for service selection situation-similarity, was undertaken. This is discussed in the next chapter.

Chapter 11

Relaxing the Test for Service Selection Situation-Similarity

In this chapter, the need for relaxing the test for service selection situation-similarity is explained. The associated concept of similar situation clusters (SSCs) is subsequently introduced, and a procedure for identifying such clusters defined. An evaluation of the use of SSCs in the design-adhering recommending registry prototype is then made.

11.1 Why Relax the Situation-Similarity Test?

As stated in Section 9.3.1, the second aspect of the basic CF-based recommending registry design which required further investigation and improvement was the test for service selection situation-similarity. This is one of the tests used in identifying the relevant service selections on which a consumer's personalised service recommendation is based. To recap, in terms of the basic design, relevant service selections are those in the service selection history which pass the three tests of type-matching, recency and situation-similarity detailed in Section 8.3.1. A service selection is type-matching if it refers to a service of the type specified in the consumer's service request, is recent if it was made within the type-specific time-window, and is situation-similar if, for the type-specific situation attribute subset, every situation attribute value exactly matches that of the consumer.

The nature of this basic test for service selection situation-similarity is a direct consequence of the symbol-interpreted, and thus generally applicable, formats for the service selection history and a requesting consumer's situation attributes specified in Sections

8.2.1 and 8.2.2; in terms of comparing the history-recorded situation attributes of a service selection against those of the consumer, the only operation that can be applied to an attribute value pair is an equality test. However, the test is an overly strict one. The symbol-based format of a service selection history essentially causes all service selections of a particular type to be interpreted as being made in a number of distinct situations, where a situation corresponds to a unique combination of attribute values for the type-specific situation attribute subset. This unique combination of subset situation attribute values will be referred to as a situation "tuple". With the basic situation-similarity test, therefore, only recent type-matching service selections with a situation tuple exactly matching that of the consumer, which were thus made in an identical situation, will be identified as relevant. However, what of those service selections with non-matching tuples, which were made in non-identical situations? Some of these other situations could be considered similar to the identical situation in terms of how service appropriateness is perceived. Consequently, many of the service selections made in these situations could also be appropriate selections for the requesting consumer, and would therefore be of relevance in generating the recommendation. This point relates back to the original research question of how to identify situation-similar service selections, detailed in Section 6.2.1.

Imagine that the prototype registry had actually been deployed in the DCS printer scenario to recommend printers to requesting consumers, with the registry configured to use the printer-specific situation attribute subset of Location & Role, and that a newly-arrived researcher submits a request for a printer from room F141 of the Computing Science department. With the basic design, the relevant service selections identified by the prototype registry would consist only of those recent printer selections made in an identical situation to that of the researcher, with a situation tuple of [Location = "f141", Role = "researcher"]. However, some of the recent printer selections made in other non-identical situations might also be of relevance in generating the printer recommendation, such as those with situation tuple [Location = "f141", Role = "post_grad"] or [Location = "g141", Role = "researcher"]. The first situation tuple refers to those recent printer selections made by post-graduate students in the same room as the researcher, whilst the second refers to those made by researchers in a nearby room. Given that printer appropriateness is likely to be perceived by people in these situations in a similar manner to how it would be perceived in the requesting researcher's situation, many of these printer selections could also be appropriate selections for the researcher himself.

"Similar Situation Clusters" and their Benefits

Ideally, therefore, a recommending registry should respond to a consumer's service request by identifying those recent type-matching service selections which were made not only in an identical situation to that of the consumer, but also in all those other situations where service appropriateness is perceived in a similar manner. This set of similar situations, including the identical situation itself, will be referred to as the "similar situation cluster" (SSC) of the requesting consumer's situation. Basing a personalised service recommendation on the relevant service selections made in the SSC, rather than on only those made in the consumer's identical situation, potentially offers several benefits.

Firstly, since a recommendation should be based on more information, it could be more effective. Supplied with more evidence in the form of more service selections, a recommendation generation algorithm should be better able to determine the collective perceived appropriateness of each available type-matching service.

Secondly, since relevant service selections would be obtained from a cluster of multiple situations, rather than a single situation, there should be less chance of no relevant service selections being identified in response to a consumer's service request. Thus, a more effective, proper recommendation could be generated for the consumer, rather than one that simply consisted of all the available type-matching services in a single tied rank.

Finally, since a recommendation should be based on more information, it should be less affected by spamming, assuming that the consensus-based recommendation generation algorithm (CB) is used. As was discussed in the last chapter, the aim of using the CB algorithm was to absorb the opinion of spamming individuals into that of the consensus opinion of all those who made relevant service selections. Since a larger number of individuals would presumably be responsible for the relevant service selections made in the SSC, the opinion of any spamming individual would be more diluted in the calculated consensus preference ranking, and thus the consequent recommendation should be less affected by his spamming.

Thus, the use of SSCs should, theoretically, enable a recommending registry to be more effective under both normal and spamming conditions.

11.2 Realising the Concept of SSCs

In view of the perceived benefits of basing a personalised service recommendation on those relevant service selections made in the SSC of the requesting consumer's situation, I investigated how the concept could be integrated into the recommending registry design. Essentially, the main issue centred on how such type-specific SSCs could be identified for situations within a recommending registry scenario. Only once a registry is configured with these SSCs can it generate personalised service recommendations that are based on this more relaxed notion of service selection situation-similarity.

11.2.1 SSC Identification Defined

The problem of SSC identification can be summed up as follows. Consider a recommending registry which is being deployed in a particular scenario. For each service type t recommended, the registry has been configured by the developer to use a particular situation attribute subset in assessing service selection situation-similarity. In the service selection history, the service selections of type t are therefore interpreted by the registry as being made in a set of distinct scenario situations S, where a situation corresponds to a unique tuple of attribute values for the t-specific situation attribute subset. For each of these situations $s \in S$, the corresponding similar situation cluster (SSC) needs to be identified. This cluster would consist of the subset of the scenario situations S where, for service type t, the perception of service appropriateness is similar to that in situation s. The subset obviously contains s itself.

How then can the SSC of each situation s be identified? The most basic solution would be to require the developer himself to identify each cluster, based on his knowledge and understanding of the different scenario situations and the service type concerned. However, this would be very time-consuming for the developer, and, moreover, the identified clusters would be based solely on his subjective view. An uninformed developer might consider two situations to be similar in terms of perceived service appropriateness but, in reality, they might be completely dissimilar. Thus, a bad SSC for a situation could actually cause a recommendation generated for a requesting consumer in that situation to be *less* effective than if the cluster was not used.

Given these negative points, the ideal solution to SSC identification is one in which the developer has little or no involvement, and in which the identified clusters are based

on rather more than fallible human intuition concerning the similarity of situations. The approach I developed in this research satisfies both criteria, and involves SSCs being identified primarily by the recommending registry itself. On initial assessment, it might appear that it would be very difficult for a registry to identify those situations that comprise a cluster, given that this would involve comparing situations for similarity in terms of perceived service appropriateness, when situations are interpreted by the registry only as symbol tuples. Returning to the example of the prototype registry given earlier, consider how the (printer-specific) SSC would be identified for the situation (tuple) of [Location = "f141", Role = "researcher"]. To the informed human eye, the other situation of [Location = "g141", Role = "researcher"] could immediately be identified as being similar, and therefore part of the cluster; the situations consist of physically nearby locations and identical roles, so people in these situations are likely to perceive departmental printer appropriateness in a similar manner. However, to the registry, these situations show no similarity at all as their tuples do not match exactly.

The approach I devised to enable a recommending registry to identify the SSC of each situation side-steps this problem. No attempt is made to assess the similarity of situations in terms of their symbol-interpreted tuples, instead similarity is assessed through comparison of the *recommendations* generated for each situation. A recommendation generated in response to a type t service request in any given situation will be based on the recent type t service selections of people in that situation, who made those selections according to their perceptions of service appropriateness. Thus, a recommendation provides an up-to-date, tangible representation of how people have collectively perceived service appropriateness in a given situation. Similar recommendations should therefore inherently imply similar views of service appropriateness.

Given this, I devised the following novel approach to enable a recommending registry to identify the SSC of a situation s. Firstly, the registry generates a recommendation r for service type t in situation s. Then, the equivalent recommendations of all other situations are compared against r for mathematical similarity. All those situations with recommendations similar to r are consequently considered similar to s in terms of perceived service appropriateness, and are therefore identified as the SSC of s. Since each of these situations has been demonstrated as being similar to s in the recent past, it is assumed that any type t service selections made there in the present and the near future could also be appropriate selections for a consumer in s, and would therefore be of relevance in

generating a type t recommendation for him.

In more precise terms, the identification of SSCs for service type t consists of the following three steps.

1. Situation Identification and Recommendation Generation: Through assessment of type t service selections in the service selection history, the registry identifies all scenario situations S. Next, the registry generates a type t recommendation for each situation $s \in S$. Assuming that the consensus-based recommendation generation algorithm is used, the fourth and final algorithm step (see Section 10.4.1) is not applied. To recap, this step involves ensuring that a recommendation contains only those services that are currently available. Given that a recommendation is being used as a recent representation of collectively perceived service appropriateness within a situation, and not to select an actual service, current service availability is irrelevant. For similar reasons, if no relevant service selections are identified, then no recommendation is generated for s, as there is no evidence on which to base a representation of service appropriateness.

Once this is done, S is filtered to contain only those situations for which recommendations were generated; the filtered version of S will be referred to as S'. Given that this approach to SSC identification is based on the comparison of recommendations, it cannot be applied to those situations with no recommendation. Thus, the SSC of such a situation is considered to contain only the situation itself.

2. Situation Recommendation Comparison: Let r_s be the generated recommendation for a situation $s \in S'$. For every pair of situations $\{i, j\} \in S'$, the mathematical similarity between r_i and r_j is then calculated.

As each situation recommendation is a ranking of services (ordered by appropriateness), the problem of recommendation similarity can be seen as an instance of the mathematical problem of ranking similarity. This problem has been considered within the statistical context of rank correlation [64], and the two main ranking distance measures developed to address it, the Kendall tau and Spearman footrule metrics, are of relevance in calculating situation recommendation similarity. However, the original metrics assumed that compared rankings were permutations, in that each ranking consisted of the same set of elements in a total order, i.e. no ties. As such, they cannot be directly used to calculate situation recommendation similarity because a) compared situation recommendations may not all contain the same services, as only services selected in a situation are ranked; and

b) compared situation recommendations may be partial orders, containing ties.

However, generalised versions of both the Kendall tau and Spearman footrule metrics have recently been developed by Fagin et al [39], in the field of meta-search, to address similar issues to those discussed, and these are applicable. In line with Fagin, before any comparison for situation recommendation similarity is made, all recommendations are modified to ensure that they rank the same set of services, i.e. the union U of those contained in *all* recommendations. Essentially, every recommendation is assessed, and those union services which are not currently contained within it are subsequently placed in a bottom tied rank.

Fagin et al have shown [39] that the generalised versions of the Kendall tau and Spearman footrule metrics are mathematically equivalent (the original metrics were shown to be equivalent by Diaconis and Graham [32]). In view of this, only the generalised Kendall tau metric, referred to as K^{gen}, is used in this research.

For every pair of situations $\{i, j\} \in S'$, the mathematical similarity between their corresponding, recommendations r_i and r_j is therefore calculated using K^{gen} as follows[1]. Recollect that, once modified, every situation recommendation ranks all services in U. Let $P = \{\{x, y\} \mid x \neq y \text{ and } x, y \in U\}$ be the set of unordered pairs of distinct services. Also, let $r_s(x)$ be the rank of a service x in recommendation r_s.

Then to calculate $K^{gen}(r_i, r_j)$, the *Kendall distance* between r_i and r_j, the *Kendall penalty* for every pair $\{x, y\} \in P$ of distinct services of U is first calculated according to one of the following three cases:

- In both r_i and r_j, x and y are in different ranks - If x and y are in the same order in r_i and r_j (e.g. $r_i(x) > r_i(y)$ and $r_j(x) > r_j(y)$), then let the Kendall penalty $k^{gen}_{x,y}(r_i, r_j) = 0$. If x and y are in the opposite order in r_i and r_j (e.g. $r_i(x) > r_i(y)$ and $r_j(x) < r_j(y)$), then let $k^{gen}_{x,y}(r_i, r_j) = 1$.

- In both r_i and r_j, x and y are tied for the same rank (i.e. $r_i(x) = r_i(y)$ and $r_j(x) = r_j(y)$) - Let $k^{gen}_{x,y}(r_i, r_j) = 0$.

- In one of the recommendations r_i and r_j, x and y are tied for the same rank; in the other, x and y are in different ranks (e.g. $r_i(x) = r_i(y)$ and $r_j(x) > r_j(y)$) - Let $k^{gen}_{x,y}(r_i, r_j) = 0.5$.

[1] Although K^{gen} can calculate the similarity between rankings of any type of element, given its specific use in this research, it will be defined here in terms of the situation recommendations of services.

The Kendall distance between r_i and r_j is defined as the sum of Kendall penalties for all service pairs. That is:

$$K^{gen}(r_i, r_j) = \sum_{\{x,y\} \in P} k^{gen}_{x,y}(r_i, r_j) \quad (11.1)$$

Essentially, the Kendall distance will measure the (dis)similarity between situation recommendations in terms of (dis)agreement over the ordering of service pairs. The distance between two identical recommendations (e.g. r_i and r_i) will be 0, as the recommendations agree on the ordering of every service pair. The largest distance will occur between two recommendations where one is the reverse of the other (e.g. A < B < C < D and D < C < B < A, where "<" means ranked above); the recommendations disagree on the ordering of every service pair.

Figure 11.1 provides an example of the Kendall distances being calculated between the corresponding recommendations of five situations $S' = \{S1, S2, S3, S4, S5\}$. The actual situation recommendations are shown at the top of the figure, and each one has been modified to ensure that it contains the same services $U = \{A, B, C, D\}$. There are six pairs of distinct services $P = \{\{A, B\}, \{A, C\}, \{A, D\}, \{B, C\}, \{B, D\}, \{C, D\}\}$. Thus, calculation of a Kendall distance involves calculating, and then summing, the Kendall penalties for each of these six service pairs. Each row in the bottom table shows the calculated Kendall penalties and Kendall distance for a pair of situation recommendations. The "Basic" row entry shows the Kendall distance in its original form; the "Norm" row entry shows it in its normalised form, which ranges from 0 (completely similar) to 1 (completely dissimilar). The normalised form is obtained by dividing the original distance by the largest possible Kendall distance (6 in this example), which occurs when two situation recommendations disagree on the ordering of every service pair.

Note that the normalised Kendall distance between r_{S1} and r_{S4} is 0, since they are identical, and the normalised distance between r_{S1} and r_{S3} is 1, since one is the reverse of the other.

For brevity, the normalised Kendall distance between recommendations r_i and r_j may be referred to as $K^{gen}_{norm}(r_i, r_j)$ in future.

3. SSC Determination: Finally, for each situation $s \in S'$, the Kendall distances between its recommendation and those of all other situations (including itself) are assessed to determine those situations which are similar to s. The SSC of s is then set to be these similar situations.

Rank	r_{S1}	r_{S2}	r_{S3}	r_{S4}	r_{S5}
1	A	B	D	A	D
2	B	A	C	B	B
3	C	C, D	B	C	C
4	D		A	D	A

$\{i,j\} \in S'$	$k^{gen}_{x,y}(r_i, r_j), \{x,y\} \in P$						$K^{gen}(r_i, r_j)$	
	{A,B}	{A,C}	{A,D}	{B,C}	{B,D}	{C,D}	Basic	Norm
{S1,S1}	0	0	0	0	0	0	0	0
{S1,S2}	1	0	0	0	0	0.5	1.5	0.25
{S1,S3}	1	1	1	1	1	1	6	1
{S1,S4}	0	0	0	0	0	0	0	0
{S1,S5}	1	1	1	0	1	1	5	0.83
{S2,S2}	0	0	0	0	0	0	0	0
{S2,S3}	0	1	1	1	1	0.5	4.5	0.75
{S2,S4}	1	0	0	0	0	0.5	1.5	0.25
{S2,S5}	0	1	1	0	1	0.5	3.5	0.58
{S3,S3}	0	0	0	0	0	0	0	0
{S3,S4}	1	1	1	1	1	1	6	1
{S3,S5}	0	0	0	1	0	0	1	0.17
{S4,S4}	0	0	0	0	0	0	0	0
{S4,S5}	1	1	1	0	1	1	5	0.83
{S5,S5}	0	0	0	0	0	0	0	0

Figure 11.1: Calculating Kendall Distances between Situation Recommendations

The obvious approach to identifying such situations is for the developer himself to define a maximum normalised Kendall distance d. If the distance between a situation's recommendation and that of s is less than or equal to d, then the situation is considered similar to s and therefore part of s's similar situation cluster. If d were set to 0.25 in the previous example, then the SSC of S1 would be {S1, S2, S4}, since $K^{gen}_{norm}(r_{S1}, r_{S1}) = 0$ (inevitably), $K^{gen}_{norm}(r_{S1}, r_{S2}) = 0.25$, and $K^{gen}_{norm}(r_{S1}, r_{S4}) = 0$. S3 and S5 would not be considered similar to S1, as their corresponding normalised Kendall distances are greater than 0.25 (1 and 0.83 respectively).

However, although this approach is relatively simple, the required involvement of the developer might not necessarily lead to the best SSCs being identified. Essentially, similarity between situations in terms of recommendation Kendall distance is being equated to similarity in terms of perceived service appropriateness. Ideally, the developer should pick the value of d that causes the corresponding SSC of each situation $s \in S'$ to contain all those situations that are truly similar to s in terms of perceived service appropriateness, and no more. For this "ideal d", the effectiveness of a recommending registry should be maximised. Unfortunately, with only intuition to direct him, a developer might not necessarily pick ideal d. If he sets d too low, then the cluster of a situation s might not contain certain, truly similar situations. If he sets d too high, then the cluster of a situation s might contain certain, dissimilar situations. Either way, recommending registry effectiveness will suffer.

Consequently, the approach I devised in this research does not involve the developer. Rather, through a process of directed experimentation and evaluation, the recommending registry itself attempts to identify the ideal d for each situation s that leads to the best SSC for s, and which therefore maximises registry effectiveness. Precise details of this self-optimisation procedure are given in the next chapter.

11.2.2 Using SSCs in a Deployed Recommending Registry

In order for a recommending registry to be able to generate personalised service recommendations based on the relevant service selections made in the SSC of a requesting consumer's situation, it must first be constructed to identify type-specific SSCs according to the approach defined above. Then, in deploying the recommending registry in its scenario, the developer must instruct the registry to identify the SSCs of scenario situations, for each service type recommended. For a particular service type, an "SSC mapping" should be

constructed, which maps each situation to its corresponding SSC.

Once configured in this way, the registry must operate as follows when generating a personalised service recommendation in response to a type t service request made by a consumer in situation s (recollect that a situation corresponds to a unique tuple of values for the t-specific situation attribute subset). As normal, the relevance of each history-recorded service selection is ascertained using the three tests of type-matching, recency and situation-similarity. The tests of type-matching and recency are still those of the basic registry design specified in Section 8.3.1. However, the situation-similarity test is now a more relaxed version: a service selection is situation-similar if its (t-specific) situation tuple exactly matches that of *any* situation in the t-specific SSC of situation s. The registry can apply this test having obtained the SSC of s from the SSC mapping for type t. Once all relevant service selections have been identified, a personalised service recommendation is generated in the normal fashion using a recommendation generation algorithm, and then returned to the consumer.

It should be noted that the recommending registry must be able to identify relevant service selections using either the strict or relaxed versions of the situation-similarity test. The relaxed version will be used when normal recommendations are being generated for consumers, but the original strict version is still required when the registry is generating situation recommendations during the SSC identification process.

Finally, the developer may wish to instruct the recommending registry to perform SSC identification periodically during its deployment lifetime. This should ensure that the identified type-specific SSCs remain accurate, reflecting how service appropriateness is currently perceived in different scenario situations.

11.3 Evaluation of SSCs

To determine whether the use of SSCs did indeed enable a recommending registry to be more effective under both normal and spamming conditions, two experiments were performed. The experiments take a similar form to those of the last two chapters, with different configurations of the prototype recommending registry being evaluated for effectiveness under both normal and simulated spamming conditions using the ESSE-based scheme of Chapter 7. The same DCS printer scenario ESSE set was used, consisting of the 10397 printer selections identified between 04/01/2004 00:00 and 31/01/2004 00:00 from

the recorded departmental service selection history, with the ESSE condition defined as $m = 5$ and $n = 28$ days. Thus, the results of the following experiments can be compared against those of the previous two chapters.

The registry prototype was modified to enable it to identify type-specific SSCs according to the approach defined in Section 11.2.1, with a type-specific SSC mapping taking the form of a hash-table. The Relevant Service Selection Obtainer component of the prototype (see Figure 9.2) was also modified, to enable relevant service selections to be identified using either the strict or relaxed versions of the situation-similarity test. To recap, the Obtainer component constructs a data-request in response to a consumer's service request that retrieves the relevant service selections from the service selection history, in the form of their ServiceID, UserID and WhenOccurred attributes. Since the history is stored in the MySQL database table, the request is an SQL SELECT statement, with the three service selection relevance tests of type-matching, recency and situation-similarity being specified as a conjunction of conditions in the WHERE clause. Thus, to enable the relaxed situation-similarity test to be used, the Obtainer component was modified to respond to a consumer's service request by first obtaining the type-specific SSC of the consumer's situation from the corresponding SSC mapping, and then constructing the data-request SELECT statement with a WHERE condition that required the situation tuple of a service selection to match that of any SSC situation.

For example, imagine that a consumer submitted a printer request, at 10/08/2004 14:25:32, with situation attributes of:

```
HourOfDay = "14"
Location = "f141"
Role = "researcher"
```

Also imagine that, in recommending departmental printers, the prototype registry was configured to use a time-window of 8 weeks and a situation attribute subset of Location & Role, and that the SSC of the consumer's situation tuple [Location = "f141", Role = "researcher"] was:

```
[Location = "f141" Role = "researcher"]
[Location = "g101" Role = "post_grad"]
[Location = "g141" Role = "researcher"]
[Location = "f141" Role = "post_grad"]
```

```
[Location = "f142" Role = "post_grad"]
```

Then the constructed SELECT statement would be:

```
SELECT ServiceID,UserID,WhenOccurred FROM ServiceSelectionHistory
WHERE (ServiceType = "Printer")
AND (((Location = "f141") AND (Role = "researcher"))
OR ((Location = "g101") AND (Role = "post_grad"))
OR ((Location = "g141") AND (Role = "researcher"))
OR ((Location = "f141") AND (Role = "post_grad"))
OR ((Location = "f142") AND (Role = "post_grad")))
AND (WhenOccurred >= "2004-06-15 14:25:32")
```

Since the primary aspect of investigation was the use of SSCs, all experiments undertaken involved the prototype registry being alternatively configured without, and then with, SSCs. Effectiveness results could then be compared to assess the impact of SSC usage. As in the last chapter, the prototype registry was configured to use a time-window of 12 weeks. It was also configured to use the consensus-based recommendation generation algorithm of InfLK-InfMC$_1$, with RPVoter (*new_service_offset* = 5 and *old_service_offset* = 5); the registry had previously proved most effective when using this CB variation (see Section 10.6). However, for similar reasons to those described in the last chapter, it was decided to vary the situation attribute subset in each experiment. By investigating SSC usage for all eight different printer-specific subsets of HourOfDay, Location and Role, behaviour noted across most subsets seemed likely to be a general trend. Clearly, no SSCs can be identified when the subset of No Attributes is used, given that all history-recorded printer selections will be interpreted as being made in the same situation.

When the prototype registry was configured to use SSCs, the (printer-specific) SSCs consisted of those that would have been identified if SSC identification had been performed at 04/01/2004 00:00, just before any of the evaluated ESSEs occurred. Using the self-optimisation procedure described in the next chapter, SSCs were chosen for all scenario situations that maximised prototype registry effectiveness for the future time period during which the evaluated ESSEs occurred (04/01/2004 00:00 - 31/01/2004 00:00). It should be noted that this is an idealised arrangement, which was utilised purely to assess the potential capability of SSC usage in maximising recommending registry effectiveness.

11.3.1 Experiment One - SSCs Under Normal Conditions

The aim of the first experiment was to determine whether the use of SSCs could enable a recommending registry to be more effective under normal conditions.

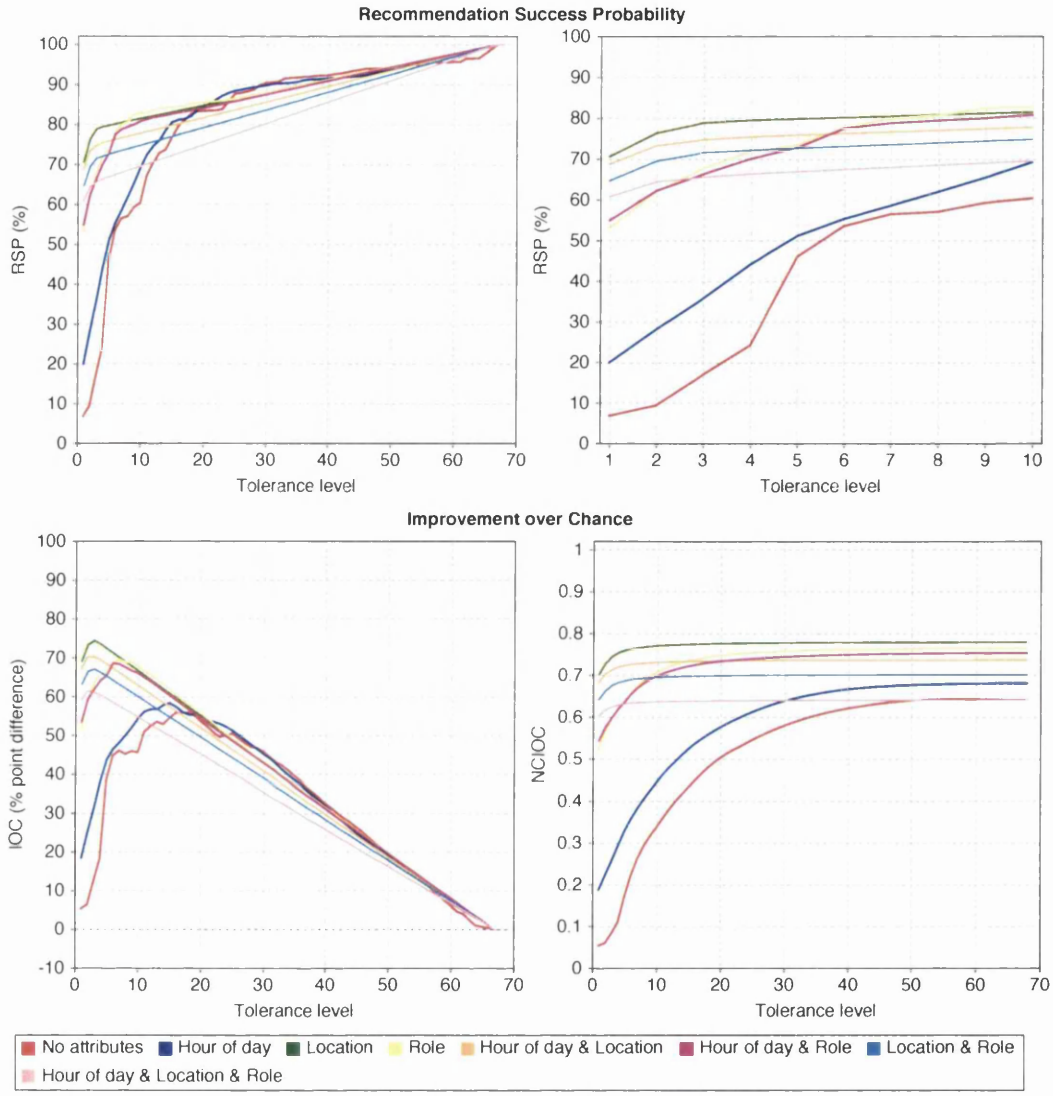
Setup

The prototype registry was configured to use, and evaluated for, each situation attribute subset of HourOfDay, Location and Role, with and without SSCs. Those eight prototype configurations in which SSCs were not used will be referred to as the "basic" configurations, whilst the corresponding eight in which SSCs were used will be referred to as the "SSC-using" configurations. The evaluation results of the basic configurations are shown in Figure 11.2 (these are the same as in Figure 10.16), whilst those of the SSC-using configurations are shown in Figure 11.3.

Results

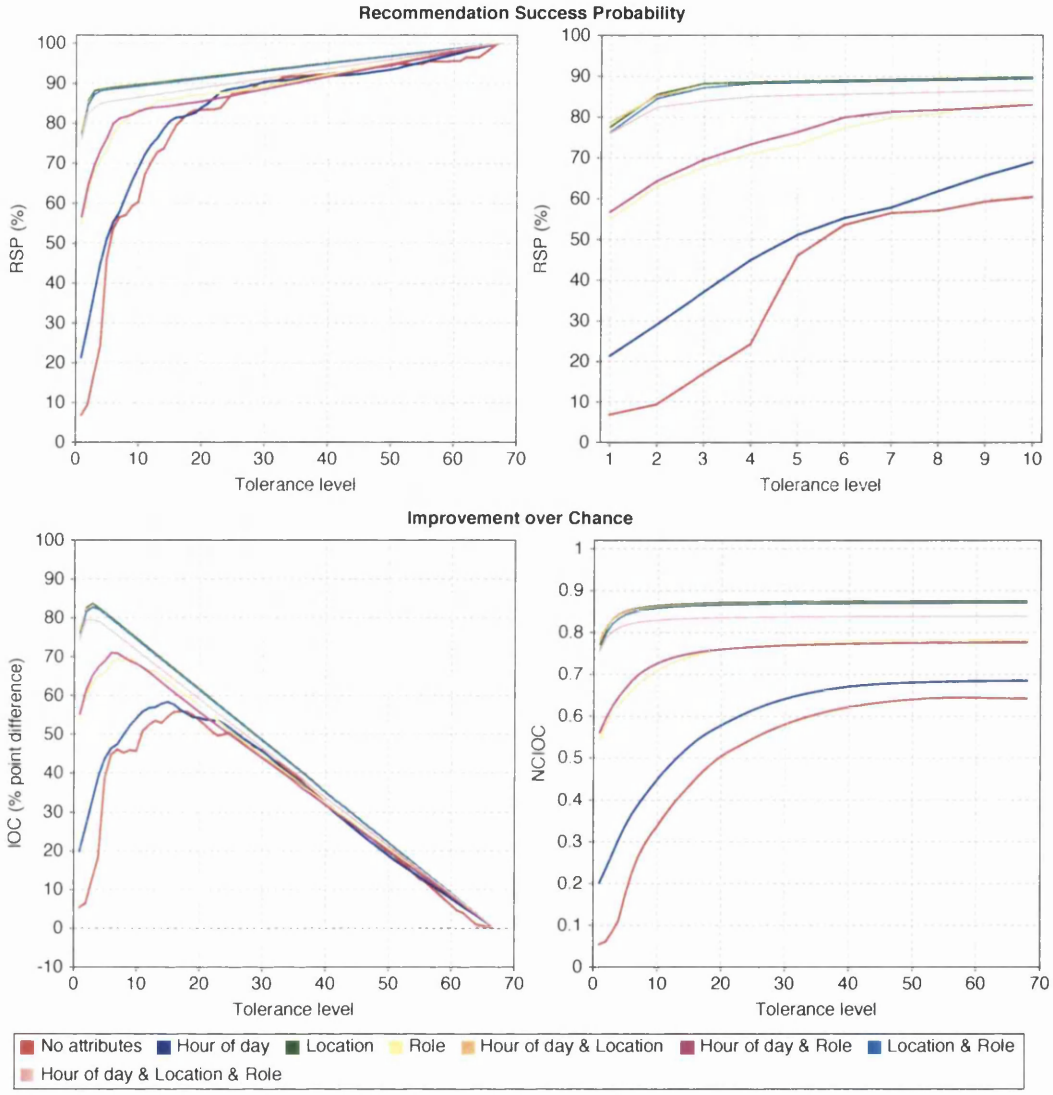
A distinct trend can be identified by comparing each pair of corresponding basic and SSC-using configurations which use the same situation attribute subset (apart from the No Attributes pair). The trend is that, of the two configurations, the SSC-using configuration is consistently more effective than the basic one. For example, in the case of Location, the NCIOC value of the SSC-using configuration is 0.8509, an increase of 0.0915 over the basic configuration (NCIOC = 0.7594). Indeed, the largest increase is 0.1859, from 0.6316 to 0.8175, which occurred for the HourOfDay & Location & Role pair.

Discussion Presumably, this improvement in effectiveness through the use of SSCs can be attributed to the reasons outlined at the end of Section 11.1. Firstly, it was asserted that since recommendations should be based on a greater number of relevant service selections, they should be more effective. This would appear to be demonstrated in the evaluation results. Recollect that, in the results table, the "Success Recommendations" cell of a configuration row refers to the number of ESSEs for which proper, "successful" recommendations were generated, which were based on a non-zero number of relevant service selections. The "Successful Recommendations Info" cells provide information about the relevant service selections on which each successful recommendation was based: the average number of relevant service selections, the average number of selecting individuals responsible for them, and the average number of services they refer to. For each pair of



Situation Attribute Subset	Recommendations			Successful Recommendations Info			Cumul. IOC (level 5)	NCIOC (level 5)	Rank
	Success	Failure		Mean # Users	Mean # Services Selected	Mean # Service Selections			
		No service selections	No user situation						
No attributes	10397	0	0	868.54	64.96	39342.84	81.18	0.17	8
Hour of day	10397	0	0	460.67	45.83	3257.31	156.84	0.3284	7
Location	8737	1419	241	63.56	5.29	3449.91	362.71	0.7594	1
Role	10360	0	37	75	13.96	4865.22	305.61	0.6399	4
Hour of day & Location	8380	1776	241	42.7	2.83	324.84	345.67	0.7237	2
Hour of day & Role	10268	92	37	51.31	8.32	422.77	303.98	0.6365	5
Location & Role	7824	2332	241	64.32	4.1	3522.77	328.1	0.6869	3
Hour of day & Location & Role	7288	2868	241	46.07	2.62	349.51	301.68	0.6316	6

Figure 11.2: Experiment One - Without SSCs



Situation Attribute Subset	Recommendations			Successful Recommendations Info			Cumul. IOC (level 5)	NCIOC (level 5)	Rank
	Success	Failure		Mean # Users	Mean # Services Selected	Mean # Service Selections			
		No service selections	No user situation						
No attributes	10397	0	0	868.54	64.96	39342.84	81.18	0.17	8
Hour of day	10397	0	0	499.42	46.58	4104.44	161.19	0.3375	7
Location	9924	232	241	63.39	5.55	3631.68	406.43	0.8509	1
Role	10360	0	37	89.39	14.87	5641.07	307.97	0.6448	6
Hour of day & Location	9575	581	241	61.61	4.63	2014.9	405.32	0.8486	2
Hour of day & Role	10324	36	37	71.11	9.63	1630.23	318.02	0.6658	5
Location & Role	9682	474	241	66.05	4.89	3848.8	402.11	0.8419	3
Hour of day & Location & Role	9112	1044	241	71.97	5	2570.96	390.45	0.8175	4

Figure 11.3: Experiment One - With SSCs

corresponding configurations, a successful recommendation generated by the SSC-using configuration was indeed based on a larger (average) number of relevant service selections than one generated by the basic configuration. Now consider the configuration pair for Role. For both configurations, the same number of successful recommendations were generated, but the SSC-using configuration has a higher NCIOC value (0.6448 versus 0.6399). A similar result is also shown for the HourOfDay configuration pair, with the same number of successful recommendations being generated, but the SSC-using configuration being more effective (0.3375 versus 0.3284). Thus, the greater amount of evidence in the form of more relevant service selections would appear to enable a SSC-using recommending registry to generate more effective recommendations. As will be seen, further corroborating evidence is also shown in Experiment Two, when, to assess the prototype registry under spamming conditions, corresponding basic and SSC-using configurations were evaluated for a trimmed ESSE set for which only successful recommendations were generated. Under non-spamming conditions ($n = 0$), for every configuration pair, the SSC-using configuration is more effective than the basic one.

Secondly, it was asserted that since relevant service selections would be obtained from a cluster of SSC multiple situations, rather than a single situation, there should be less chance of no relevant service selections being identified in response to consumers' service requests. Thus, a greater number of more effective, proper recommendations could be generated for consumers, rather than those that simply consisted of all available type-matching services in a single tied rank. This is clearly demonstrated in the evaluation results by considering the configuration pairs for any of the situation attribute subsets involving Location. For example, in the case of Location & Role, 7824 proper recommendations were generated using the basic configuration, but 9682 were generated with the SSC-using configuration, an increase of 1858 (23.75%). Similarly, in the case of HourOfDay & Location & Role, SSC usage caused the number of proper recommendations generated to improve from 7288 to 9112, an increase of 1824 (25.03%).

Conclusion

This experiment has demonstrated that the use of SSCs does indeed enable a recommending registry to be more effective under normal conditions.

11.3.2 Experiment Two - SSCs Under Spamming Conditions

The aim of the second experiment was to determine whether the use of SSCs could enable a recommending registry to be more effective under spamming conditions.

Setup and Presentation

Setup Various configurations of the prototype registry, which differed in whether SSCs were used or not, were evaluated under a variety of simulated spamming conditions, and the effectiveness results compared.

This experiment is very similar to Experiment Three of the last chapter (see Section 10.5.3). To recap, that was concerned with investigating how the consensus-based recommendation generation algorithm (CB) performed in the face of orchestrated spamming. Thus, it involved various prototype registry configurations being evaluated under simulated conditions where the registry was being (collectively) spammed in every situation by a number of individuals. This experiment takes a very similar form, but is concerned with SSC usage, rather than with the CB algorithm.

The prototype registry was assessed in terms of 7 sub-experiments. For each sub-experiment, the registry was configured to use a different situation attribute subset of HourOfDay, Location and Role (but not No Attributes). It was then configured with and without SSCs, and these two basic and SSC-using configurations were evaluated under simulated spamming conditions in the same manner as in Experiment Three: see Section 10.5.3. To recap, each configuration was evaluated in terms of the recommendations it generated in response to the ESSEs of the "trimmed" successful ESSE set. This set consisted of only those ESSEs for which proper recommendations could be generated under normal conditions (without the use of SSCs), which were based on a non-zero number of relevant service selections. In evaluating a configuration, for each of these ESSEs, the relevant service selections identified during the recommendation process were intercepted, and then augmented with a set of simulated sham service selections deliberately designed to boost the recommendation ranks of the corresponding 10 "bottom" services. To recap, the bottom services of the ESSE were those that would be worst-ranked in a recommendation generated using the SCO algorithm under normal conditions (without the use of SSCs). The added sham selections were simulated as being made by n individuals, and the particular values of n will be discussed below. The ESSE recommendation generated by the evaluated configuration was then based on this augmented set of relevant service

selections.

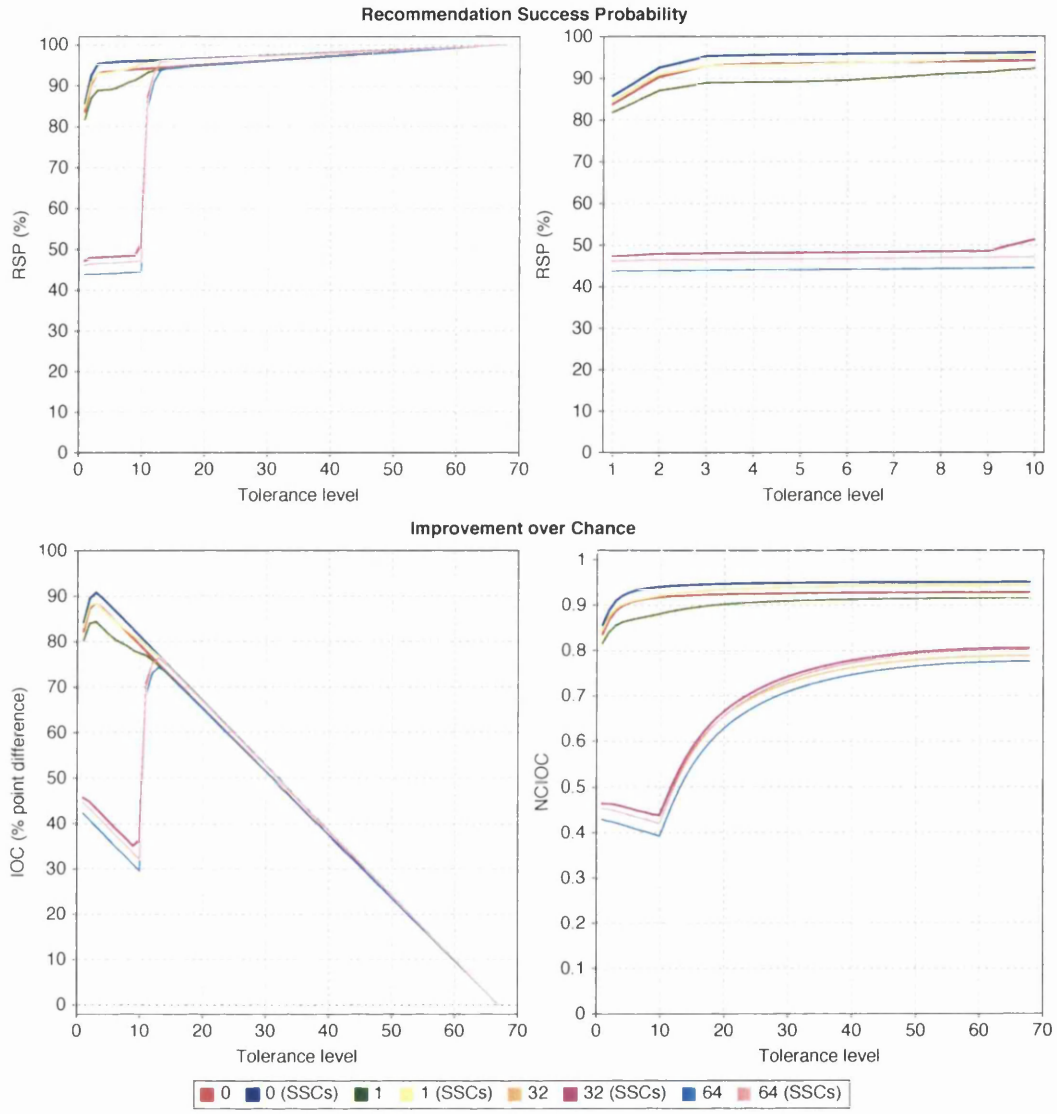
In overall terms, therefore, the prototype registry was simulated as being spammed in every situation in which an ESSE occurred. For a particular situation, the registry was spammed for 10 particular services by n individuals. Note that the spam services may have differed between situations. Both the basic and SSC-using registry configurations were evaluated in this way for 4 different values of n. The first 2 values were 0 (no spamming) and 1. The other 2 values were calculated from the average number of selecting individuals responsible for the relevant service selections on which an ESSE recommendation was based (without the use of SSCs) under normal non-spamming conditions (i.e. $n = 0$). If this average number of individuals is i, then n was set to $\frac{i}{2}$ and i.

Presentation For each sub-experiment, the evaluation results for the basic and SSC-using configurations under the 4 different simulated spamming conditions are presented in the standard graph/table format. The results for the Location and HourOfDay & Location sub-experiments are given in Figures 11.4 and 11.5 respectively; the results for the other 5 sub-experiments are given in Appendix D, in Figures D.1 to D.5. In a particular sub-experiment, the evaluation details of the SSC-using configuration for a particular number of spamming individuals n are referred to as "n (SSCs)". The evaluation details of the basic configuration for the same number of n spamming individuals are simply referred to as "n".

Results

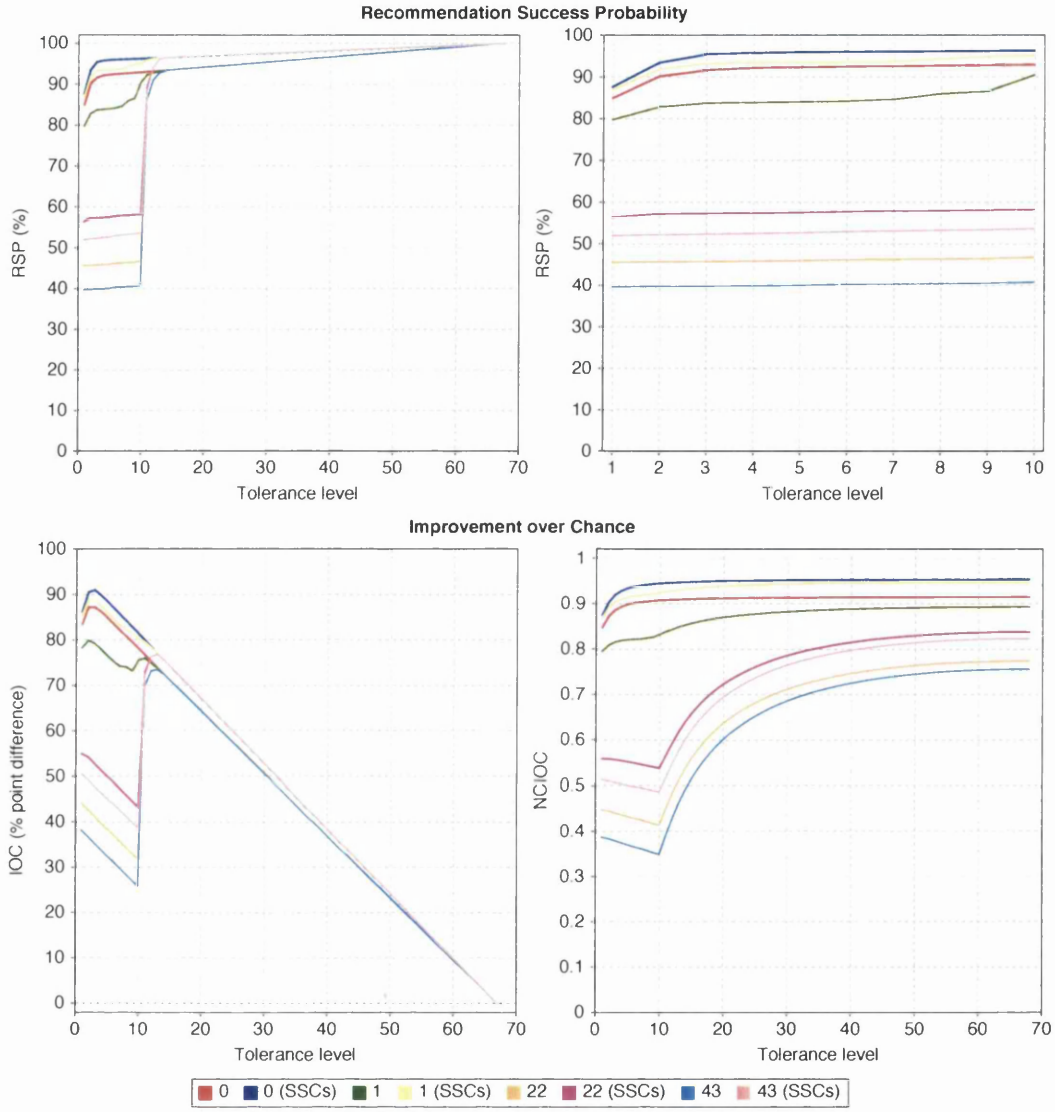
A distinct trend can be identified from the 7 sub-experiments, by comparing the basic and SSC-using configurations for any value of n. The trend is that, of the two configurations, the SSC-using configuration is consistently more effective than the basic one. For example, in the case of HourOfDay & Location, the NCIOC value of the SSC-using configuration at $n = 1$ is 0.9145, an increase of 0.0941 over the basic configuration (NCIOC = 0.8204). Similarly, at $n = 22$, the increase is 0.1191 (from 0.4322 to 0.5513), and 0.1310 at $n = 43$ (from 0.3699 to 0.5009). Indeed the largest increase is 0.1578, from 0.7715 to 0.9273, for HourOfDay & Location & Role at $n = 1$.

Discussion Presumably, this improvement in effectiveness through the use of SSCs can be attributed to the reason outlined at the end of Section 11.1. It was asserted that since recommendations should be based on more relevant service selections made by



# Spamming Individuals	Recommendations			Successful Recommendations Info			Cumul. IOC (level 5)	NCIOC (level 5)	Rank
	Success	Failure		Mean # Users	Mean # Services Selected	Mean # Service Selections			
		No service selections	No user situation						
0	8737	0	0	63.56	5.29	3449.91	431.62	0.9037	3
0 (SSCs)	8737	0	0	70.96	5.83	4006.18	442.45	0.9264	1
1	8737	0	0	64.56	15.29	35714.24	413.49	0.8657	4
1 (SSCs)	8737	0	0	71.96	15.74	36270.52	432.61	0.9058	2
32	8737	0	0	95.56	15.29	1035908.61	216.89	0.4541	6
32 (SSCs)	8737	0	0	102.96	15.74	1036464.89	217.03	0.4544	5
64	8737	0	0	127.56	15.29	2068367.32	197.18	0.4128	8
64 (SSCs)	8737	0	0	134.96	15.74	2068923.59	209.99	0.4397	7

Figure 11.4: Experiment Two - Location



# Spamming Individuals	Recommendations			Successful Recommendations Info			Cumul. IOC (level 5)	NCIOC (level 5)	Rank
	Success	Failure		Mean # Users	Mean # Services Selected	Mean # Service Selections			
		No service selections	No user situation						
0	8380	0	0	42.7	2.83	324.84	428.87	0.8979	3
0 (SSCs)	8380	0	0	67.75	4.44	2185.79	445.91	0.9336	1
1	8380	0	0	43.7	12.83	3429.39	391.82	0.8204	4
1 (SSCs)	8380	0	0	68.75	14.2	5290.33	436.78	0.9145	2
22	8380	0	0	64.7	12.83	68624.84	206.43	0.4322	7
22 (SSCs)	8380	0	0	89.75	14.2	70485.78	263.34	0.5513	5
43	8380	0	0	85.7	12.83	133820.29	176.65	0.3699	8
43 (SSCs)	8380	0	0	110.75	14.2	135681.24	239.25	0.5009	6

Figure 11.5: Experiment Two - HourOfDay & Location

more individuals, they should be less affected by spamming, assuming that the consensus-based recommendation generation algorithm (CB) is used (which it was: InfLK-InfMC$_1$). This would appear to be demonstrated in the evaluation results. For any of the 7 sub-experiments, consider the average number of individuals responsible for the relevant service selections on which a successful recommendation was based (the "Mean # Users" cell), for the basic and SSC-using configurations at $n = 0$ (no spamming). The value is indeed always higher when SSC is used. For example, in the case of HourOfDay & Location (Figure 11.5), an average of 42.7 individuals were responsible for the relevant service selections with the basic configuration, but the number had increased to 67.75 individuals with the SSC-using configuration. Thus, the opinions of any spamming individuals are more diluted in a calculated CB consensus preference ranking generated using the SSC-using registry configuration, since they make up a smaller proportion of the individuals on which the ranking is based, and the consequent recommendation is more effective. For example, in the case of HourOfDay & Location at $n = 1$, an average of 2.29% (1/43.7) of the individuals on whose opinions a recommendation was based using the basic registry configuration were spamming. However, the figure had dropped to 1.45% (1/68.75) with the SSC-using configuration.

Conclusion

This experiment has demonstrated that the use of SSCs does indeed enable a recommending registry to be more effective under spamming conditions.

11.4 Conclusion

Through the two experiments performed, it has been demonstrated that the use of SSCs can enable a recommending registry to be more effective under both normal and spamming conditions. In conclusion, therefore, it can be seen that SSCs, and the corresponding relaxation in the service selection situation-similarity test, are worthwhile improvements to the CF-based recommending registry design.

11.5 Summary

The focus of this chapter has been the second improvement made to the basic CF-based recommending registry design, namely that concerned with relaxing the test for service

selection situation-similarity. Motivation for the need to relax the test has been given, followed by a discussion of the concept of similar situation clusters (SSCs) and their potential benefits.

A detailed explanation of my novel approach to the identification of SSCs has then followed, and the three steps of the approach - situation identification and recommendation generation, situation recommendation comparison, and SSC determination - have been defined. Directions concerning the use of SSCs in a deployed recommending registry have also been specified.

The final part of the chapter has detailed the two experiments that were performed to evaluate the effectiveness of a recommending registry, with and without SSCs, under normal and spamming conditions. The conclusion has been drawn that the use of SSCs can improve recommending registry effectiveness, and that the relaxation of the test for service selection situation-similarity is a second worthwhile improvement to the basic CF-based recommending registry design.

The third and final improvement to the basic design, that concerned with the developer task of registry configuration, is discussed in the next chapter.

Chapter 12

Recommending Registry Configuration using Self-Optimisation

In this chapter, a self-optimisation procedure for enabling a design-adhering recommending registry to configure itself is motivated and defined. An evaluation of this procedure is then made.

12.1 Motivation for Minimising Developer Involvement

As stated in Section 9.3.1, the third and final aspect of the CF-based recommending registry design which required further investigation and improvement was the developer task of registry configuration.

Section 8.3.2 specified that, for a design-adhering recommending registry to become operational in a scenario, the developer would first need to configure it for each service type recommended. More precisely, for each service type t, the developer would need to specify two aspects of the tests used to identify relevant service selections on which a type t recommendation would be based: the subset of t-specific situation attributes used in the service selection situation-similarity test, and the length of time-window used in the service selection recency test.

As would be expected, these type-specific configurations of a design-adhering recommending registry have a significant impact on the effectiveness of the recommendations that it generates. For example, consider how the basic prototype registry performed in

the DCS printer scenario for the different printer-specific configurations evaluated in the Chapter 9 experiments. When the registry was configured to use the situation attribute subset of Location and a time-window of 8 weeks in generating printer recommendations, it had an NCIOC value of 0.7543 (see Figure 9.6). However, when configured to use the situation attribute subset of HourOfDay and a time-window of 12 weeks, it had an NCIOC value of only 0.357 (see Figure 9.5).

Clearly, although a design-adhering recommending registry has the *potential* to generate significantly effective recommendations, its actual effectiveness will depend on how well it has been configured for each service type. Presumably, a developer would attempt to configure a registry so as to maximise effectiveness, choosing the particular situation attribute subset and length of time-window for each service type t that he considered would generate the most effective type t recommendations. A developer might diligently evaluate different configurations for type t in a manner similar to that of the Chapter 9 experiments, and choose the configuration that had performed best in the past. Alternatively, he might simply choose the configuration for type t that he intuitively believed would perform best, without such time-consuming investigation. However, regardless of the approach taken, the very fact that a developer is required to use considerable personal judgement when configuring the registry could lead to ill advised choices being made, and thus less effective recommendations being generated.

Moreover, if the relaxed notion of service selection situation-similarity, as defined in the previous chapter, was utilised in the recommending registry, then a further configuration burden could be placed on the developer. With SSCs being identified for type t according to the process defined in Section 11.2.1, a developer would be required to pick the ideal maximum normalised Kendall distance (ideal d) for each situation, in order to obtain the best corresponding SSCs. As was noted in the previous chapter, the developer's involvement could lead to inappropriate SSCs being obtained for situations.

12.2 Registry Self-Optimisation

In view of the fact that such developer involvement in registry configuration could lead to registry under-performance, I devised a novel, alternative approach to configuration. With this refined approach, for each service type t, the recommending registry itself evaluates different configurations and selects the one which performed best. Theoretically, a type t

configuration that would have enabled the registry to generate the most effective type t recommendations in the recent past should also enable effective recommendations to be generated in the present and the near future. Such registry self-optimisation minimises developer involvement, and should, hopefully, maximise registry effectiveness.

12.2.1 The Self-Optimisation Procedure Defined

The registry self-optimisation procedure for selecting the configuration for service type t can be summarised as follows. Firstly, the registry identifies a set of ESSEs from its service selection history that refer to services of type t, and that occurred within a time-segment that began at some point in the recent past (such as a number of weeks ago) and ended in the present, at the point at which self-optimisation is occurring. Next, for a number of different type t configurations, the registry evaluates itself in terms of this recent ESSE set using the evaluation scheme defined in Chapter 7. Specifically, it configures itself in a number of different ways for service type t, in terms of the situation attribute subset, the length of time-window and the calculated SSCs, and for each configuration, evaluates itself in terms of the type t recommendations it generates in response to the (type t) ESSEs. Finally, the registry compares the different configurations in terms of their corresponding NCIOC values (for a particular tolerance level), and chooses the one with the highest value. In other words, the registry configures itself in such a manner that would have enabled it to generate the most effective type t recommendations in the recent past. Since the registry's "performance" in the recent past is being evaluated in terms of the identified ESSE set, the set should be representative of typical type t service requests that might be made within the deployment scenario.

As a clarifying example, imagine that the prototype registry is actually being deployed in the DCS printer scenario to recommend departmental printers to requesting consumers. Deployment occurs on 04/01/2004. Consequently, the registry self-optimisation procedure is run at 04/01/2004 00:00, in order for the registry to configure itself for the printer service type. The registry identifies an ESSE set which consists of departmental printer selections that occurred in, say, the last 4 weeks, from 07/12/2003 00:00 to 04/01/2004 00:00. Next, the registry evaluates itself in terms of this representative ESSE set for different configurations. The configurations differ in terms of the subset of the printer-specific situation attributes (HourOfDay, Location and Role), the length of time window (e.g. 1 hour, 1 day, 1 week, 2 weeks, 4 weeks, 8 weeks, 12 weeks ...) and the calculated

SSCs. Finally, the registry selects the configuration with the highest NCIOC value. When deployed, the registry will generate printer recommendations using this configuration.

In more precise terms, the registry self-optimisation procedure for selecting the configuration for service type t consists of the following two steps.

1. ESSE Set Identification:

The recent representative ESSE set E of type t service selections is identified by the recommending registry from its service selection history. The ESSEs will be drawn from a time-segment that begins at some point in the recent past and ends in the present, at the moment of self-optimisation. The developer must specify the length of this time-segment, from which the start point can then be calculated. He must also specify the ESSE condition, (m and n as explained in Section 7.4.2). ESSE set E will be identified according to these developer-specified values.

For example, in the prototype registry example of printer self-optimisation occurring at 04/01/2004 00:00, E could have been identified with a time-segment of 4 weeks (start point is then 07/12/2003 00:00), $m = 5$ printer selections and $n = 28$ days.

2. Configuration Evaluation and Comparison:

The registry evaluates itself in terms of ESSE set E for a number of different type t configurations, and selects the one for which it performs best, in terms of NCIOC value at a particular tolerance level. The developer must specify this tolerance level.

Type t configurations can differ in terms of the subset of t-specific situation attributes, the length of time-window, and the calculated SSCs. Thus, there are an almost infinite number of possible configurations. There may be a finite number of situation attribute subsets, but the time-window length could be anything between 1 second and 1 year or more, and in the SSC identification process, the maximum normalised Kendall distance d for each situation could be anything between 0 and 1, leading to different calculated SSCs. Given this, to remain feasible, the self-optimisation procedure does not evaluate every possible type t configuration, but a finite number.

The developer is required to specify a set B of time-window lengths which he considers are sensible values that could lead to effective type t recommendations (i.e. not a tiny time-window of 10 seconds). Let A be the set of all possible subsets of the t-specific situation attributes. Then the registry evaluates itself only for those configurations with a situation

attribute subset $a \in A$ and a time-window length $b \in B$. For a particular attribute subset / time-window length pair (a, b), only one configuration is evaluated. Rather than assess all possible (a, b) configurations that differ in terms of calculated SSCs, the registry first identifies the (a, b) configuration with the best SSCs that maximise registry effectiveness for ESSE set E. Only this "best (a, b)" configuration is evaluated; the process used to identify this configuration will be explained later. Thus, in total, a maximum of $|A| * |B|$ configurations could be evaluated.

Ideally, all $|A| * |B|$ configurations would be evaluated by the registry and the best performing one selected. However, if $|A|$ or $|B|$ were large, then the evaluation of all of these configurations could take a considerable amount of time. Thus, the self-optimisation procedure was devised to evaluate only a *subset* of these configurations, that should still enable the best performing configuration to be identified, or at least one that performed almost as well.

The particular procedure devised was based on the observation from Figures 9.5 and 9.6 that the choice of situation attribute subset had a much greater impact on registry effectiveness than the length of time-window. Consequently, it was decided that configurations that differed in terms of situation attribute subset should be evaluated first, before those that differed in terms of time-window length. More precisely, the developer is required to specify which time-window length in B he considers should lead to the most effective type t recommendations; this will be referred to as b^{start}. Then, the registry evaluates itself for those $|A|$ configurations that differ in terms of situation attribute subset, but have this specified time-window length. The most effective e of these $|A|$ configurations is identified. Next, the registry evaluates itself for those $|B| - 1$ configurations that have the same situation attribute subset as e, but different time-window lengths. At this point, the most effective of these $|B| - 1$ configurations and e is identified. If e still remains the most effective, the self-optimisation procedure terminates, and the registry configures itself to use e. However, if e is not the most effective, but one of the $|B| - 1$ configurations with alternative time-window length b^{alt} is, this indicates that the registry is more effective when configured to use b^{alt}. Thus, the whole process is repeated using b^{alt}. That is, those $|A|$ configurations that differ in terms of situation attribute subset, but have this length of time-window b^{alt} are evaluated, followed by the identification of a new e, evaluation of those with time-window lengths different to e (i.e. not b^{alt}), and possible termination with e selected. In the worst case, all $|A| * |B|$ configurations will be evaluated. However, in

the best case, when the self-optimisation procedure terminates on the first iteration with e selected, only $|A| + |B| - 1$ configurations will have been evaluated.

Example

Returning to the prototype registry example of printer self-optimisation, there are 3 printer-specific situation attributes: HourOfDay, Location and Role; for brevity, these will be referred to as "H", "L" and "R" respectively. A therefore consists of 8 attribute subsets: {{No attributes}, {H}, {L}, {R}, {H, L}, {H, R}, {L, R}, {H, L, R}}. Imagine that B is specified as {4 weeks, 8 weeks, 12 weeks}. Then, there are 24 (8 * 3) different printer configurations that could be evaluated. Imagine that b^{start} is specified as 12 weeks. Then, the self-optimisation procedure might operate as follows. Firstly, the 8 configurations with time-window length of 12 weeks are evaluated:

1. No attributes / 12 weeks
2. H / 12 weeks
3. L / 12 weeks
4. R / 12 weeks
5. H, L / 12 weeks
6. H, R / 12 weeks
7. L, R / 12 weeks
8. H, L, R / 12 weeks

Next, e is identified: the configuration with the highest NCIOC value. Imagine that e is H, L, R / 12 weeks. Then, the 2 other configurations with the same situation attribute subset as e but different time-window lengths are evaluated:

9. H, L, R / 4 weeks
10. H, L, R / 8 weeks

Finally, the most effective of these 2 configurations and e is identified. If e still remains the most effective, the self-optimisation procedure terminates with e selected, after only 10 configuration evaluations. However, if one of the other 2 configurations is most effective instead, with a time-window length of b^{alt}, the process is repeated using b^{alt}. For example,

if H, L, R / 8 weeks is most effective, the process would be repeated with a time-window length of 8 weeks. Obviously, in this further iteration, if a configuration has been evaluated before (e.g. H, L, R / 8 weeks), it can be skipped over, and its previously calculated NCIOC value used in the comparison of configuration effectiveness.

How is the Best (a, b) Configuration Identified?

As was noted earlier, a type t configuration with situation attribute subset $a \in A$ and time-window length $b \in B$ that is evaluated by the recommending registry in step 2 of the self-optimisation procedure (e.g. H, L, R / 12 weeks in the previous example) is the one that was identified as having the best SSCs that maximise registry effectiveness for the ESSE set E identified in step 1. How is this best (a, b) configuration identified? Essentially, the SSC identification process defined in the previous chapter is used (see Section 11.2.1). Firstly, the registry configures itself for type t with situation attribute subset a and time-window length b. Next, the steps in the SSC identification for type t are followed. To recap, in step 1 of the SSC identification process, through assessment of the type t service selections in its service selection history, the registry identifies all scenario situations S; each situation corresponds to a unique tuple of attribute values for the situation attribute subset a. An attempt is then made to generate a recommendation for each situation $s \in S$, and S is filtered to contain only those situations for which recommendations were actually generated; this filtered set is referred to as S'. In step 2, for every pair of situations in S', the Kendall distance between their corresponding recommendations is calculated. Finally, in step 3, for each situation $s \in S'$, the Kendall distances between its recommendation and those of all other situations (including itself) are assessed to determine those situations which are similar to s. The SSC of s is set to be these similar situations.

In describing step 3 of the SSC identification process (Section 11.2.1), it was explained that a situation would be considered similar to s, and therefore part of the SSC of s, if the normalised Kendall distance between the situation's recommendation and that of s was less than a defined maximum distance d. The challenge is to set d at the ideal level that leads to the best SSC for s being obtained, which maximises recommending registry effectiveness. The description in the previous chapter only noted that the registry picks this value of d itself. Complete details of this sub-process will now be given.

Firstly, the registry ranks all situations in S', ordering them by the Kendall distance between their recommendations and that of s, smallest to largest. This ranked list l_s may

contain ties, as some situations may have the same Kendall distance to s. Secondly, the registry assesses E, and identifies those ESSEs that were made in situation s; this ESSE subset will be referred to as E^s. Thirdly, the registry configures itself with an SSC for s that consists only of those situations in the first rank of l_s (the "nearest" situations). It then evaluates itself in terms of E^s, noting the NCIOC value. Next, the registry configures itself with an enlarged SSC for s that consists of those situations in the first *and second* ranks of l_s. Again, it evaluates itself in terms of E^s, and compares the obtained NCIOC value against that of the smaller SSC from the last iteration. If the NCIOC value is greater than or equal to this last value, then the SSC is again enlarged to also contain those situations in the next rank of l_s, and again evaluated. However, if the NCIOC value is smaller than this last value, the sub-process stops, and the SSC of s is set to be the smaller one from the last iteration. In other words, the SSC of s is being enlarged to contain more situations from more ranks in l_s, as long as registry effectiveness increases (or remains constant). When the process stops, either because registry effectiveness deteriorated with SSC enlargement, or the SSC could not be enlarged any further (there are no more ranks in l_s), the resulting SSC maximises recommending registry effectiveness for E^s; conceptually, d has been set to a value that results in this SSC. Consequently, by applying this sub-process to every $s \in S'$, the resulting calculated SSCs maximise registry effectiveness for E as a whole. If a situation s is not represented in E (i.e. $|E^s| = 0$), its SSC is set to contain only the situation itself; there are no ESSEs with which to assess the impact of enlarging the SSC. Thus, the best (a, b) configuration (with the best SSCs) has been identified, and can now be evaluated by the registry. With the explanation of this step 3 sub-process, the SSC identification process has now been defined completely.

Example As a clarifying example, consider Figure 11.1 of the previous chapter again. This showed the recommendations generated in step 1 of the SSC identification process for five situations $S' = \{$S1, S2, S3, S4, S5$\}$. It also showed the Kendall distances calculated between every pair of situation recommendations in step 2. Now consider how step 3 would occur, according to the sub-process defined above. Figure 12.1 shows the ranked list l_s for each $s \in S'$, which contains all the situations ordered by the Kendall distance between their recommendations and that of s. For each rank of l_s, the normalised Kendall distance of the situations in that rank is shown in brackets. Now consider how the best SSC would be obtained for S1. Firstly, the registry configures itself with an SSC of { S1, S4 } for S1,

since these situations are in the first rank of l_{S1}, and then evaluates itself in terms of E^{S1}. Next, the registry configures itself with an enlarged SSC of { S1, S4, S2 }, the situations in the top 2 ranks, and again evaluates itself. The NCIOC value obtained for this SSC is compared against that of the previous SSC. Assuming that the value is greater than the previous one, the registry configures itself with an enlarged SSC of { S1, S4, S2, S5 }, the situations in the top 3 ranks, and evaluation occurs again. This time, however, let us assume that the NCIOC value of this SSC is less than that of the previous SSC. Thus, the sub-process stops, and the SSC of S1 is set to { S1, S4, S2 }. Conceptually, d has been set to 0.25. This same sub-process is applied to each $s \in S'$, and the best SSCs are therefore calculated that maximise recommending registry effectiveness for E as a whole.

Rank	S1	S2	S3	S4	S5
1	S1, S4 (0)	S2 (0)	S3 (0)	S1, S4 (0)	S5 (0)
2	S2 (0.25)	S1, S4 (0.25)	S5 (0.17)	S2 (0.25)	S3 (0.17)
3	S5 (0.83)	S5 (0.58)	S2 (0.75)	S5 (0.83)	S2 (0.58)
4	S3 (1)	S3 (0.75)	S1, S4 (1)	S3 (1)	S1, S4 (0.83)

Figure 12.1: Situations Ordered By Kendall Distance

Addendum to Section 11.3 As a brief side issue, the experimental setup used in evaluating the SSC concept in the previous chapter (Section 11.3) can now be explained fully. The two experiments performed (Sections 11.3.1 and 11.3.2) involved the prototype registry being evaluated for the standard set of ESSEs identified from between 04/01/2004 00:00 and 31/01/2004 00:00. The prototype registry was evaluated for a number of different printer-specific configurations, which differed in terms of situation attribute subset. When the registry was configured to use SSCs, the (printer-specific) SSCs consisted of those that would have been identified if SSC identification had been performed at 04/01/2004 00:00, just before any of the evaluated ESSEs occurred. Using the complete SSC identification process (including the just-detailed step 3 sub-process), for a considered printer-specific configuration, the best SSCs were identified for all scenario situations that maximised prototype registry effectiveness for the future time period during which the evaluated ESSEs occurred (04/01/2004 00:00 - 31/01/2004 00:00). This was done by performing

the step 3 sub-process not with a recent representative ESSE set (e.g. the last 4 weeks, from 07/12/2003 00:00 to 04/01/2004 00:00) as normal, but with the *future* ESSE set being used for evaluation purposes itself. This was an idealised arrangement, which was utilised purely to assess the potential capability of SSC usage in maximising recommending registry effectiveness.

12.2.2 Using the Self-Optimisation Procedure

A design-adhering recommending registry should be constructed to configure itself according to the self-optimisation procedure defined above. Note that the SSC identification process defined in the last chapter is now essentially a sub-element of this procedure. Then, in deploying the recommending registry in its scenario, the developer must instruct the registry to configure itself for each service type recommended. To summarise, in instructing the registry to configure itself for service type t, the developer must specify:

- The length of the recent time-segment from which the representative ESSE set E will be identified (e.g. the last w weeks).
- The ESSE condition (m and n).
- The particular tolerance level at which the different evaluated type t configurations will be compared, in terms of NCIOC value.
- The set B of time-window lengths that type t configurations can have.
- The time-window length b^{start} in B that the developer considers should lead to the most effective type t recommendations.

The developer may also wish to instruct the recommending registry to perform self-configuration periodically during its deployment lifetime. This should ensure that the chosen type-specific configurations reflect any change that has occurred, and that registry effectiveness is maintained.

12.3 Evaluation of the Self-Optimisation Procedure

To determine whether the self-optimisation procedure did indeed enable a recommending registry to configure itself well, and thus achieve a high level of registry effectiveness, two

experiments were performed. As previously, the experiments involve the prototype recommending registry, which was modified to enable it to use the self-optimisation procedure. Each experiment involves the registry configuring itself in the DCS printer scenario at a particular point in time, and then being evaluated for NCIOC effectiveness in terms of an ESSE set identified from a time-segment that occurred just after this point. Thus, both experiments show how effective the registry would have been if it had configured itself (for the printer service type) according to the self-optimisation procedure.

As in the last chapter, the prototype registry was set to use the consensus-based recommendation generation algorithm of InkLK-InfMC$_1$, with RPVoter ($new\_service\_service$ = 5 and $old\_service\_offset$ = 5). In both experiments, the registry was instructed to configure itself for the printer service type according to the following values:

- The length of the recent time-segment from which ESSE set E was identified was 4 weeks. This length was chosen because the standard representative ESSE set used for evaluation purposes in the last three chapters had been identified from a 4 week time-segment.

- The ESSE condition was specified as m = 5 printer selections and n = 28 days. This condition was chosen because the standard representative ESSE set had been identified with these values.

- The NCIOC values of the different printer-specific configurations were to be compared at tolerance level 5. This level was chosen because it had been used in all previous prototype registry experiments.

- The set B of time-window lengths was { 4 weeks, 8 weeks, 12 weeks }. These lengths seemed sensible values, since they had been demonstrated to lead to effective departmental printer recommendations in Figure 9.6.

- The time-window length b^{start} in B was 12 weeks. This length had led to the most effective printer recommendations in Figure 9.6.

12.3.1 Experiment One - Self-Optimisation at 04/01/2004 00:00

Setup

In the first experiment, the prototype registry was instructed to configure itself for the printer service type as if it was 04/01/2004 00:00. Thus, the ESSE set E used by the

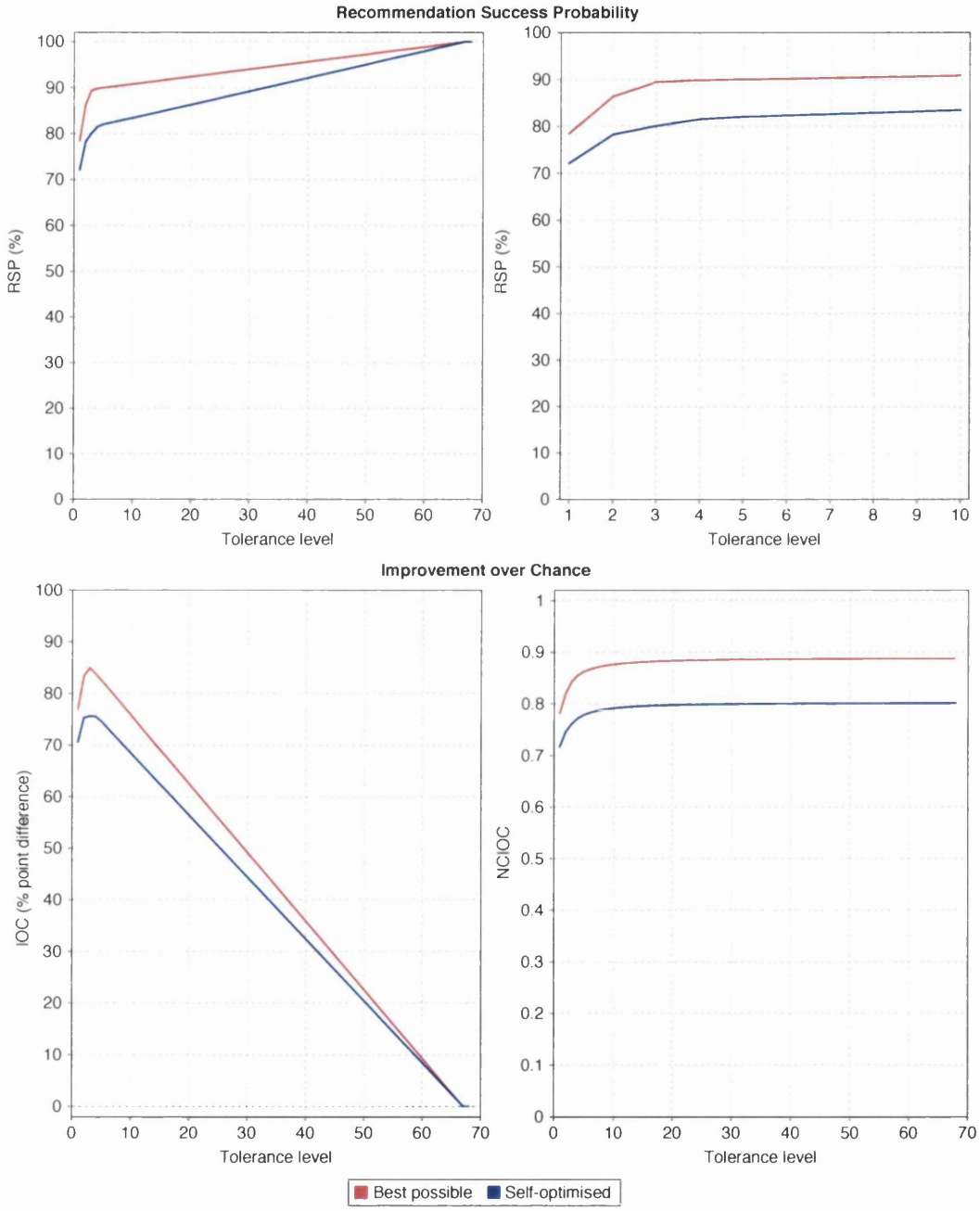
self-optimisation procedure was identified from the previous 4 week time-segment between 07/12/2003 00:00 and 04/01/2004 00:00. E consisted of 12266 printer selections involving 51 departmental printers and 604 individuals. The procedure selected the configuration with a situation attribute subset of HourOfDay & Location & Role and a time-window of 12 weeks (plus best SSCs).

The prototype registry was then evaluated for this chosen configuration in terms of the standard ESSE set used in previous chapters. To recap, this set of ESSEs consists of 10397 printer selections identified between 04/01/2004 00:00 and 31/01/2004 00:00, with the ESSE condition defined as $m = 5$ printer selections and $n = 28$ days. Thus, the registry was evaluated for the period just after self-optimisation occurred.

To provide a useful comparison, the self-optimisation procedure was also utilised to identify the best possible configuration that would have maximised prototype registry effectiveness for this evaluated ESSE set; for brevity, this set will be referred to as F. Thus, the prototype registry was again instructed to configure itself for the printer service type as if it was 04/01/2004 00:00. However, this time, the self-optimisation procedure was performed using F itself. In reality, such a self-optimisation could not occur, given that this set of ESSEs was identified from a period in the future, *after* the point in time at which the procedure was occurring. The best possible configuration selected used a situation attribute subset of Location and a time-window of 4 weeks (plus best SSCs). The registry was then evaluated for this configuration in terms of F.

Results

The evaluation results for when the prototype registry used the self-optimised configuration are shown in Figure 12.2. For comparison, the evaluation results for when the registry used the best possible configuration are also shown. The prototype registry has performed significantly well when using the self-optimised configuration: it has an NCIOC value of 0.7779. In other words, the prototype registry was almost four-fifths as effective as the perfect (and unattainable) recommending registry (i.e. NCIOC = 1). A consumer with tolerance level 1 had a 72.11% chance of finding an appropriate service. At tolerance level 2, his chances had increased to 78.25%, and 80.08% at level 3. Registry effectiveness with the self-optimised configuration is also near that obtained with the best possible configuration (NCIOC = 0.8618): the difference is 0.0869.



Self-optimised Configurations	Recommendations			Successful Recommendations Info			Cumul. IOC (level 5)	NCIOC (level 5)	Rank
	Success	Failure		Mean # Users	Mean # Services Selected	Mean # Service Selections			
		No service selections	No user situation						
Best possible	9782	374	241	41.42	3.69	672.89	411.6	0.8618	1
Self-optimised	8880	1276	241	72.14	5.01	2455.51	371.55	0.7779	2

Figure 12.2: Experiment One - Self-Optimisation at 04/01/2004 00:00

12.3.2 Experiment Two - Self-Optimisation at 04/03/2004 00:00

Setup

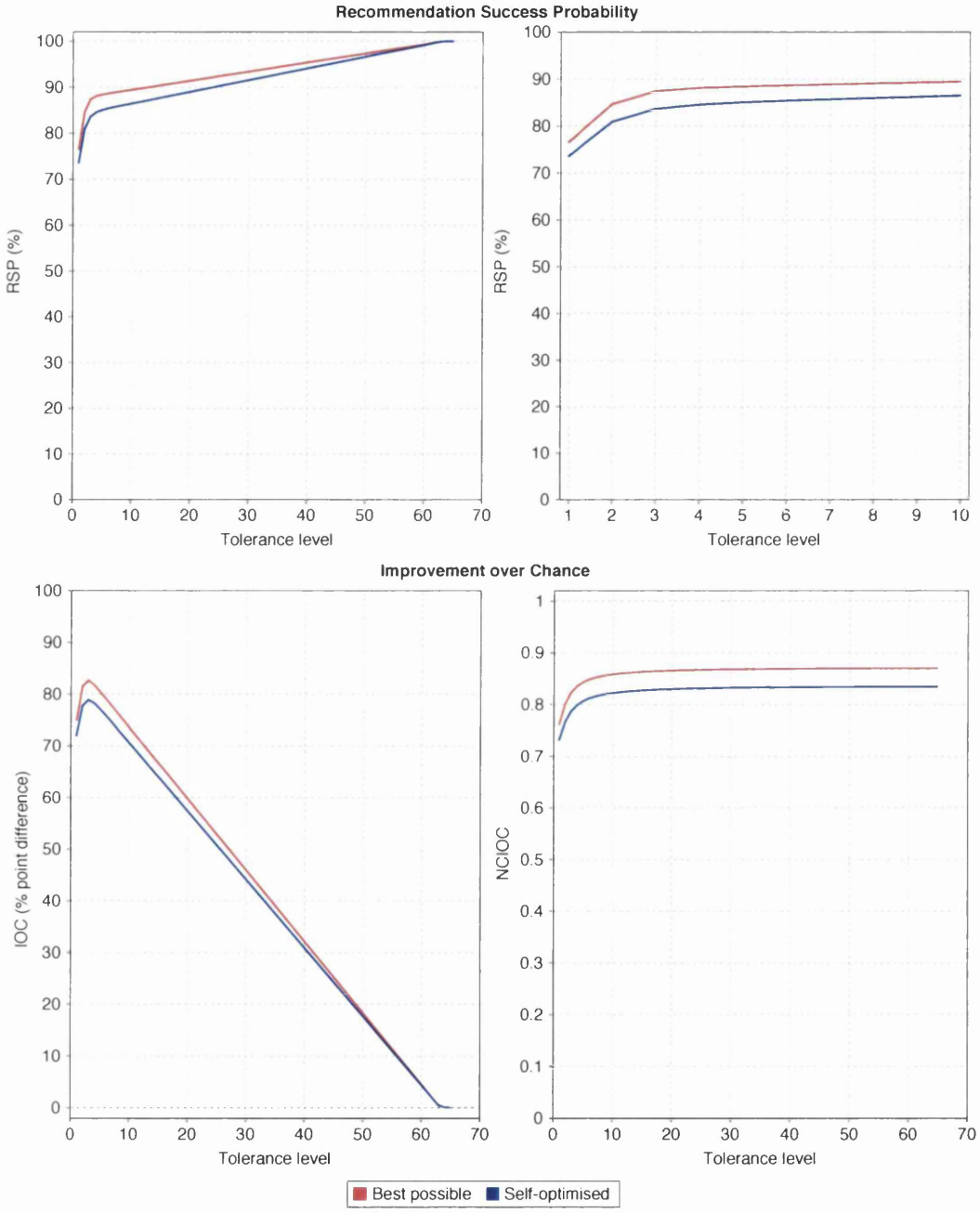
In the second experiment, the prototype registry was instructed to configure itself for the printer service type as if it was 04/03/2004 00:00, two months after the first experiment. Essentially, if it is imagined that the first registry self-optimisation occurred on the initial deployment of the prototype registry, this second self-optimisation might have been initiated to ensure that any change that might have occurred in the interim was reflected in the chosen printer-specific configuration, and that registry effectiveness was maintained. Thus, the ESSE set E used by the self-optimisation procedure was identified from the recent 4 week time-segment between 05/02/2004 00:00 and 04/03/2004 00:00. E consisted of 18421 printer selections involving 51 departmental printers and 623 individuals. The procedure selected the configuration with a situation attribute subset of HourOfDay & Location & Role and a time window of 8 weeks (plus best SSCs).

The prototype registry was then evaluated for this chosen configuration in terms of an ESSE set F that was identified from the 4 week time-segment between 04/03/2004 00:00 and 31/03/2004 00:00, with the standard ESSE condition of $m = 5$ printer selections and $n = 28$ days. This representative set of ESSEs consists of 12544 printer selections involving 54 departmental printers and 474 individuals. Thus, the registry was evaluated for the period just after self-optimisation occurred.

Again, to provide a useful comparison, the self-optimisation procedure was also utilised to identify the best possible configuration that would have maximised prototype registry effectiveness for the evaluated ESSE set F. Similar to the first experiment, the prototype registry was instructed to configure itself for the printer service type as if it was 04/03/2004 00:00, and the self-optimisation procedure was performed using F itself. The best possible configuration selected used a situation attribute subset of'Location and a time-window of 12 weeks (plus best SSCs). The registry was then evaluated for this configuration in terms of F.

Results

The evaluation results for when the prototype registry used the self-optimised configuration are shown in Figure 12.3. For comparison, the evaluation results for when the registry used the best possible configuration are also shown. Again, the prototype registry has performed



Self-optimised Configurations	Recommendations			Successful Recommendations Info			Cumul. IOC (level 5)	NCIOC (level 5)	Rank
	Success	Failure		Mean # Users	Mean # Services Selected	Mean # Service Selections			
		No service selections	No user situation						
Best possible	11401	304	839	61.6	6.1	3053.73	401.33	0.8426	1
Self-optimised	10768	937	839	60.71	4.78	1698.15	383.95	0.8061	2

Figure 12.3: Experiment Two - Self-Optimisation at 04/03/2004 00:00

significantly well when using the self-optimised configuration: it has an NCIOC value of 0.8061. This value is actually better than that which was achieved in the first experiment. A consumer with tolerance level 1 had a 73.58% chance of finding an appropriate service. At tolerance level 2, his chances had increased to 80.89%, and 83.62% at level 3. Registry effectiveness with the self-optimised configuration is again near that obtained with the best possible configuration (NCIOC = 0.8426): the difference is 0.0365.

12.4 Conclusion

Through the two experiments performed, it has been demonstrated that the self-optimisation procedure can enable a design-adhering recommending registry to configure itself well, and consequently achieve a high level of registry effectiveness. In conclusion, therefore, it can be argued that such registry self-configuration should enable a more consistently high level of registry effectiveness to be achieved than might be possible through configuration by the developer. In addition, from a developer's perspective, the task of registry configuration has been made much simpler and far less time-consuming.

12.5 Summary

The focus of this chapter has been the registry self-optimisation procedure that I devised in order to simplify the developer task of registry configuration, and to maximise registry effectiveness. The motivation for such an improvement has been given, followed by a definition of the self-optimisation procedure itself.

The main body of the chapter has been concerned with a detailed explanation of the registry self-optimisation procedure, which is based on the use of my previously-defined ESSE-based evaluation scheme, and incorporates the SSC identification process detailed in the last chapter. Information as to using the self-optimisation procedure has also been specified.

Finally, details have been given of the two experiments that were undertaken in connection with the evaluation of the self-optimisation procedure. The conclusion has been drawn that the devised registry self-optimisation procedure can enable a design-adhering recommending registry to configure itself so as to achieve a high level of registry effectiveness, and as such should simplify the developer task of registry configuration.

Now that the third and final improvement to the basic CF-based recommending registry

design has be detailed, it is possible to define an advanced design, which incorporates all three improvements (from Chapters 10, 11 and 12). This advanced design is specified in the next chapter.

Chapter 13

An Advanced Recommending Registry Design

In this chapter, an advanced CF-based SDM recommending registry design is defined, together with the sequence of developer tasks that must be performed for a design-adhering registry to become operational in a particular scenario.

13.1 An Advanced Registry Design

In view of the three improvements made to the basic CF-based SDM recommending registry design, a definition can now be given of a more advanced version. This advanced CF-based design is an augmented form of my basic design (defined in Section 8.3.1), incorporating the three improvements of the consensus-based recommendation generation algorithm (Chapter 10), the SSC-based relaxation of the service selection situation-similarity test (Chapter 11), and the self-optimisation procedure for registry configuration (Chapter 12). As before, the design is based on the abstract model of a recommending registry stated in Section 4.4. It is defined as follows.

For the particular scenario in which it will be deployed, a constructed recommending registry must:

1. Have access to a service selection history for the scenario, which adheres to the standardised format defined in Section 8.2.1. The history records details of all service selections that occur within the scenario on a continual basis. To recap, the history entry of a type t service selection records four core attributes (ServiceID, UserID, WhenOccurred and ServiceType) and a number of t-specific situation attributes.

2. Be able to acquire the situation attributes of a requesting consumer. The acquired attributes must adhere to the standardised format defined in Section 8.2.2.

3. Be able to configure itself according to the self-optimisation procedure defined in Section 12.2.1; note that the SSC identification process defined in Section 11.2.1 is a sub-element of this procedure. It is assumed that, on deployment, the registry is instructed to configure itself for each scenario service type recommended. Thus, for each service type t, the registry chooses the subset of t-specific situation attributes used in the service selection situation-similarity test, the length of time-window used in the service selection recency test, and constructs an SSC mapping which maps each scenario situation to its corresponding SSC.

In generating a personalised service recommendation for a requesting consumer, a constructed recommending registry must operate as follows:

1. As defined in the abstract model, the requesting consumer submits a service request to the registry, stating the required service type t and any specific attributes. The registry responds by acquiring the t-specific situation attributes of the requesting consumer. It also identifies available type t services by assessing the advertised service descriptions.

2. The registry identifies those relevant service selections that refer to a type t service, and that were recently made in a situation similar to that of the consumer. More precisely, three tests are applied to each history-recorded service selection:

   - Type-matching - The ServiceType attribute of the service selection is tested for equality against the required service type t specified in the consumer's service request. If the compared values are equal, the service selection is *type-matching.*
   - Situation-similarity - The relaxed version of this test is applied as specified in Section 11.2.2. To recap from Chapter 11, a scenario situation corresponds to a unique combination of attribute values for the t-specific situation attribute subset; the acquired subset situation attribute values of the requesting consumer specify the situation he is currently in, whilst those of the service selection specify the situation in which it was made. The service selection is *situation-similar* if its situation tuple exactly matches that of any situation in the t-specific SSC of the requesting consumer's situation. The registry can apply this

test having first obtained this SSC from the SSC mapping for type t constructed earlier.

- Recency - The WhenOccurred attribute of the service selection is tested, to determine whether the selection was made within the t-specific time-window. The date-time start of the time-window, TimeWindowStart, can be calculated by subtracting the time-window length from the current date-time. If WhenOccurred >= TimeWindowStart, then the service selection is *recent.*

The set of relevant service selections identified are those that are type-matching, situation-similar and recent.

3. The registry assesses the relevant service selections to rank the available type t services in an ordered list by collective perceived appropriateness. This list is the personalised service recommendation. The consensus-based recommendation generation algorithm (CB), as defined in Section 10.4.1, is used. Given the evaluation results of Chapter 10, InfLK-InfMC$_1$ with RPVoter would seem an ideal CB variation. If no relevant service selections were identified, a recommendation is generated that consists of all the available type t services in a single tied rank.

4. The registry returns the generated personalised service recommendation to the requesting consumer. As required by the abstract model, the recommendation is filterable to show only those type t services that also match the specified service attributes.

13.2 The Tasks of the Registry Developer

For a design-adhering recommending registry to become operational in a particular scenario, the developer must perform the following sequence of tasks:

1. Service selection history recording - The developer must construct the service selection history recording mechanism. For each type of scenario service, he must decide which situation attributes to record in a service selection history entry along with the core attributes. As was advocated in Section 8.2.1, the history recording mechanism should ideally be automated.

2. Recommending registry construction - The developer must construct a recommending registry that adheres to the design specified above.

3. Recommending registry configuration - For each scenario service type recommended, the developer must instruct the constructed registry to configure itself using the self-optimisation procedure. In doing so, he must specify the small number of variables defined in Section 12.2.2.

4. Recommending registry deployment - The developer must deploy the registry. This deployment can occur once the history recording mechanism has recorded a sizeable service selection history, on which the registry can generate personalised service recommendations. Periodically during the the registry's lifetime, the developer may wish to instruct the registry to reconfigure itself, so as to ensure that any change that has occurred is reflected in the chosen type-specific configurations, and that registry effectiveness is maintained.

NB: It should be noted that the final version of my working recommending registry prototype, evaluated in the last chapter (Chapter 12), adhered to this advanced design.

Chapter 14

Conclusion

In this final chapter, an assessment is given as to whether the completed research meets the original aim. This is followed by a discussion of the evaluation approach, a general discussion of the CF-based recommending registry design, and possible future work. Finally, a list of research contributions is given.

14.1 The Completed Research in Relation to the Research Aim and Thesis Statement

As was stated in the introduction (Chapter 1), the aim of my research was to attempt to address the following question:

> What general design for an SDM recommending registry would enable the generation of effective personalised service recommendations?

More precisely, it was claimed, in the thesis statement, that it would be possible to devise a general design which would enable the registry generation of effective recommendations based on an assessment of past service selections/usage. The main body of this thesis has detailed the investigation of such a design, and the final design itself is defined in Chapter 13. However, can this CF-based design be said to meet the demands of the inquiry?

The generality of the design can be seen from the fact that the inputs of a design-adhering recommending registry - the service selection history, and the service request and situation attributes of a requesting consumer - are represented and interpreted only as symbols. As a consequence, from the perspective of a registry, the nature of the data being processed is irrelevant. Thus, theoretically, it should be possible for a design-adhering

registry to be deployed in any scenario to recommend any type of service. Provided that the inputs did conform to the symbol-interpreted formats specified in the design, a registry would be able to generate recommendations. The design does specify that date-time should be represented and interpreted in a date-time format, but date-time is universal. It should be noted, however, that there could be scenarios in which the symbolic representation of required recommending registry inputs could be difficult. For example, it might be challenging to represent a floating-point situation attribute as a finite set of symbols. This would obviously affect registry deployment, and the question of such intricacies of symbolic representation would be a useful topic for future research.

A recommending registry which adhered to the specified CF-based design should also be able to generate effective personalised service recommendations. In this research, effectiveness has been demonstrated through the evaluation of the working design-adhering prototype registry within the DCS printer scenario. The version of the prototype registry evaluated in Chapter 12 (concerned with registry self-optimisation) was one which did adhere completely to the final design specified in Chapter 13. Therefore, the evaluation results of Experiments One and Two in Chapter 12 provide examples of how highly effective a design-adhering recommending registry could be: in Experiment One, the self-optimised registry achieved an NCIOC value of 0.7779 (see Figure 12.2), and in Experiment Two, it achieved one of 0.8061 (see Figure 12.3). It should be remembered that the final design incorporates the improvements of the consensus-based recommendation generation algorithm and similar situation clusters (SSCs). Thus, as was demonstrated in Chapters 10 and 11, a design-adhering recommending registry should also be able to remain effective in the face of spamming.

In summary, therefore, in terms of the thesis statement, it has been demonstrated that it was possible to devise a general design for an SDM recommending registry which would enable the generation of effective personalised service recommendations based on an assessment of past service selections/usage. Moreover, this CF-based design provides an answer to the original research question.

14.2 Discussion of the Evaluation Approach

The style of evaluation used throughout this research can be defined, as in Information Retrieval [119], as a system-oriented approach. A registry's effectiveness is assessed in

an automated manner in terms of recommendations generated in response to pre-defined representative service requests. A recommendation is considered effective if pre-defined appropriate services are ranked highly. This approach implicitly assumes that a registry which proves effective under these system-oriented test conditions will prove effective from the perspective of actual requesting consumers.

I decided early on in my research that I would need some method of assessing recommending registry effectiveness throughout the registry design process in order to enable me to assess solutions to the design research issues. The system-oriented evaluation scheme (see Chapter 7) that I devised was ideal for this. The automated and self-contained nature of the scheme, together with the single numeric value of the NCIOC effectiveness measure, enabled the rapid assessment, comparison and evolution of various design options. Thus, this same scheme was able to accommodate not only the investigation of variation in time-window length and choice of situation attributes, but also the investigation of various recommendation generation and SSC algorithms, under both normal and spamming conditions.

A more consumer/user-oriented approach was considered, but discounted. Such an approach could have consisted of users interacting with a deployed design-adhering recommending registry and providing feedback on whether the generated recommendations aided them in their service selections. However, a significant amount of time and effort would have been needed to organise, execute and complete a user study, and to collate and interpret the study results. This would have precluded the rapid and frequent assessment of the design options. Moreover, multiple user studies would have been required to assess different aspects of the design (e.g. variations in time-window length, choice of situation attributes, etc.). In addition, it would have been difficult to obtain precise quantitative measurements from user studies, making it challenging to compare research solutions (e.g. which of 28 different recommendation generation algorithms was most effective?).

Although a user-oriented evaluation approach would not have been suitable during my investigative development of the CF-based recommending registry design, it might be applicable now that the design has been completed. As has been noted in IR [34], there is value in complementing the system-oriented evaluation of an interactive recommendation system with some form of user-oriented evaluation. From the perspective of my research, this would allow an holistic investigation into whether real people did consider a design-adhering recommending registry to be of benefit. For example, did they consider the

returned personalised service recommendations to be trustworthy and of good quality, and were the recommendations generated speedily enough? It would also allow an assessment of user-registry interaction, which in turn could lead to improvements in recommendation presentation.

14.3 Discussion of the CF-based Recommending Registry Design

In addition to the main points of generality and effectiveness, there are certain other issues relating to the CF-based recommending registry design of Chapter 13 which should be discussed.

Firstly, although this research does not present a generic mechanism to support the CF-based design, a generic mechanism which could provide an almost complete implementation of a recommending registry is entirely feasible. At its core, this registry implementation could provide the functionality of a basic service discovery mechanism, to handle the advertisement and leasing of provider-supplied service descriptions. On top of this, all the design aspects relating to recommendation generation could also be provided. In fact, the prototype registry developed for this research would need only the addition of basic SDM features to become a design-supporting generic mechanism itself.

In terms of the tasks that a developer would need to perform in order for a recommending registry to become operational in a particular scenario (see Section 13.2), such a generic mechanism would completely remove the task of registry construction (task 2). Given that the registry self-optimisation procedure minimises developer involvement in registry configuration and deployment (tasks 3 and 4), the developer could concentrate his efforts on the task of constructing the service selection history recording mechanism (task 1). This last task is predominately a scenario-specific engineering problem, although the developer would need to decide upon the type-specific situation attributes to be recorded.

Secondly, while, as has been demonstrated, the CF-based design can enable effective recommendations to be generated by a design-adhering registry, just how effective the recommendations are will be dependent on the underlying service selection history. If the history provides a complete record of service selections/usage in the registry deployment scenario, and the services chosen are, for the most part, appropriate, effective recommendations should be generated. Conversely, an incomplete history containing many inappropri-

ate service selections may cause ineffective recommendations to be generated. However, in any scenario, inappropriate services should hopefully be kept to a minimum, as individuals should be motivated to choose the services most appropriate for their circumstances.

Thirdly, as was noted in Section 6.1.3, the issue of individual privacy could be a concern in relation to the recording of people's service selections for the service selection history. However, in many scenarios, such data might already be recorded for auditing or charging purposes. Moreover, an individual may actually be quite willing for his service selection behaviour to be recorded if he can perceive a gain for himself in the form of effective personalised service recommendations. In addition, in terms of the CF-based design, the behaviour of an individual is not singled out as a generated recommendation represents a consensus opinion derived from group behaviour.

14.4 Possible Future Work

With regard to future work, there are several potentially interesting areas of investigation.

As mentioned previously in this chapter, the intricacies of symbolic representation (see Section 14.1) and user-oriented evaluation (see Section 14.2) could be investigated.

A comprehensive generic mechanism which takes the form of an almost complete implementation of a design-adhering recommending registry would be both practicable and useful, as was stated in Section 14.3, . Thus, such a generic mechanism could be constructed, with particular investigation being made into how to maximise the speed/throughput of recommendation generation and how to minimise memory usage.

Although the viability and validity of the CF-based recommending registry design has been demonstrated in the real-world DCS printer scenario, further assessment of the design in another scenario would be of use. Research into the design within the context of a different style of scenario with different types of services could uncover areas of the general design which would benefit from modification or improvement. For example, a scenario where change is a regular and expected occurrence could allow an in-depth investigation to be undertaken into how well a design-adhering recommending registry would respond in such circumstances.

The CF-based design could be further examined through the actual deployment of a design-adhering recommending registry as, although my working prototype was evaluated in the DCS printer scenario, it was not deployed for active use. One interesting research

question could be: how often should a deployed registry reconfigure itself? Another issue relates to the impact that requesting consumers' responses to recommendations generated for them in the present, in terms of the consequent service selections they make, will have on the recommendations generated by the deployed registry in the future.

One particular aspect of the CF-based design that would certainly merit investigation and improvement is how a design-adhering recommending registry responds to a consumer's service request when no relevant service selections are identified. As the design currently stands, the consumer is given a recommendation that simply consists of all the available type-matching services in a single tied rank. It would be better if, in such an instance, the consumer could be provided with a more effective recommendation.

Finally, as my research was concerned primarily with the development of the CF-based recommending registry design itself, no comprehensive investigation was made into how a service selection history would be recorded (developer task 1 in Section 13.2). Although the recording of such a history would be scenario specific, it might be possible to develop some generally applicable guidelines.

14.5 Research Contributions

The main contributions of this research are:

- A general CF-based design for an SDM recommending registry.
- An evaluation scheme which can be used to assess the effectiveness of any recommending registry that adheres to this general CF-based design.

  The devised effectiveness measures of RSP (Recommendation Success Probability), IOC (Improvement over Chance) and NCIOC (Normalised Cumulative IOC) could also be used in the evaluation of any non design-adhering recommending registry that conforms to the general abstract model defined in Section 4.4. In such a case, an alternative method of choosing representative service requests (and corresponding appropriate services) to that of the evaluation scheme would need to be used.

Other contributions of this research are:

- The demonstration of a particular style of Collaborative Filtering, that incorporates aspects of Social Choice Theory (from Economics), to aid service selection.

- The identification of the similarity between a proposed SDM recommending registry and an IR system. This could lead to further investigation of IR from the perspective of developing techniques to aid service selection.

Appendix A

Glossary of Acronyms

CB Consensus-Based recommendation generation algorithm. See Section 10.4.

CF Collaborative Filtering. See Section 6.3.

ECC Extended Condorcet Criterion. See Section 10.4.4.

ESSE Experienced Service Selection Entry. See Section 7.4.1.

InfLK Inferencing Local Kemenisation. See Section 10.4.4.

InfMC$_1$ - InfMC$_4$ Inferencing Markov Chain recommendation generation algorithms. See Section 10.4.3.

IOC Improvement Over Chance recommending registry effectiveness measure. See Section 7.4.4.

IR Information Retrieval. See Section 7.3.1.

LK Local Kemenisation. See Section 10.4.4.

MC$_1$ - MC$_4$ Markov Chain recommendation generation algorithms. See Section 10.4.3.

NCIOC Normalised Cumulative IOC recommending registry effectiveness measure. See Section 7.4.5.

RPVoter Reward Punishment Voter preference ranking algorithm. See Section 10.4.2.

RSP Recommendation Success Probability recommending registry effectiveness measure. See Section 7.4.3.

SCO Selection Count Ordering recommendation generation algorithm. See Section 8.2.6.

SCOVoter Selection Count Ordering Voter preference ranking algorithm. See Section 10.4.2.

SDM Service Discovery Mechanism. See Section 3.2.

SOA Service-Oriented Architecture. See Section 2.1.

SSC Similar Situation Cluster. See Section 11.1.

STR Selection Tied Ranking recommendation generation algorithm. See Section 8.2.6.

Appendix B

Chapter 9 Details

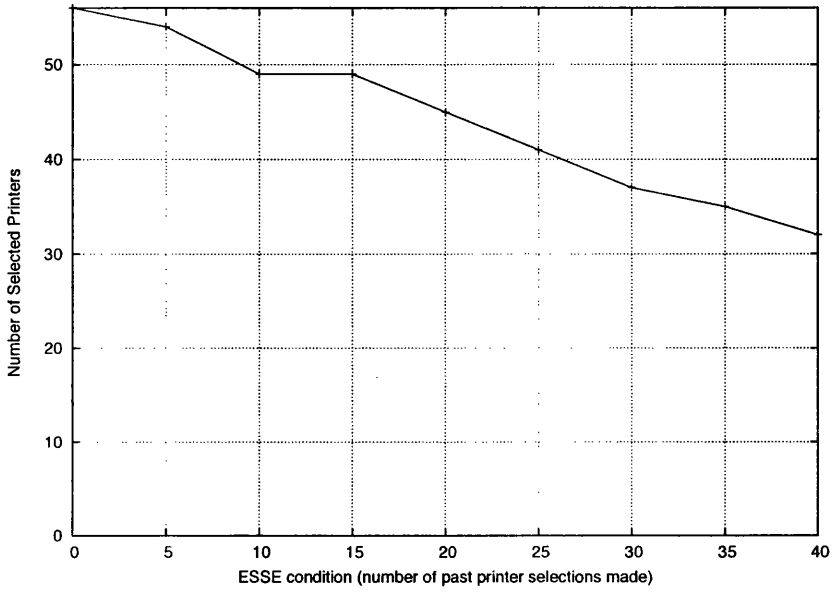


Figure B.1: The Number of Printers Referred to by an ESSE Set

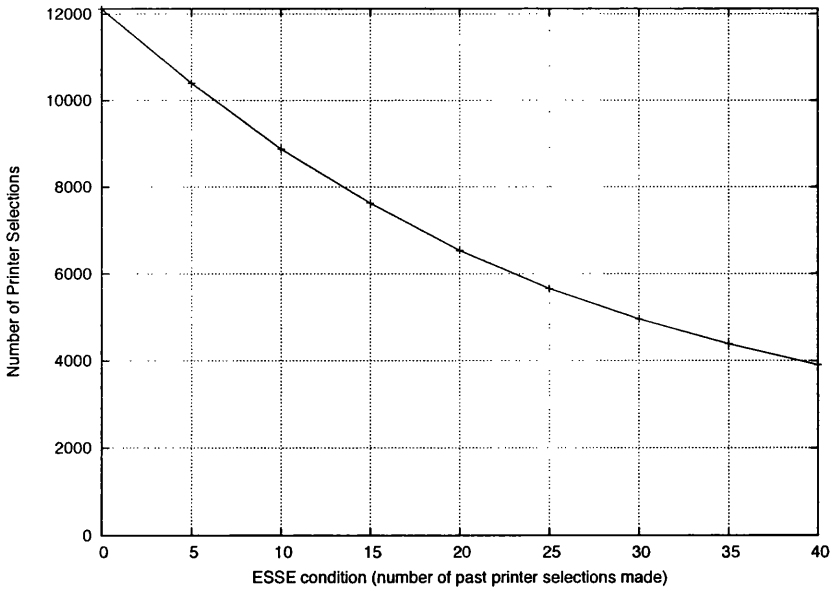


Figure B.2: The Size of An ESSE Set (Number of Printer Selections)

Appendix C

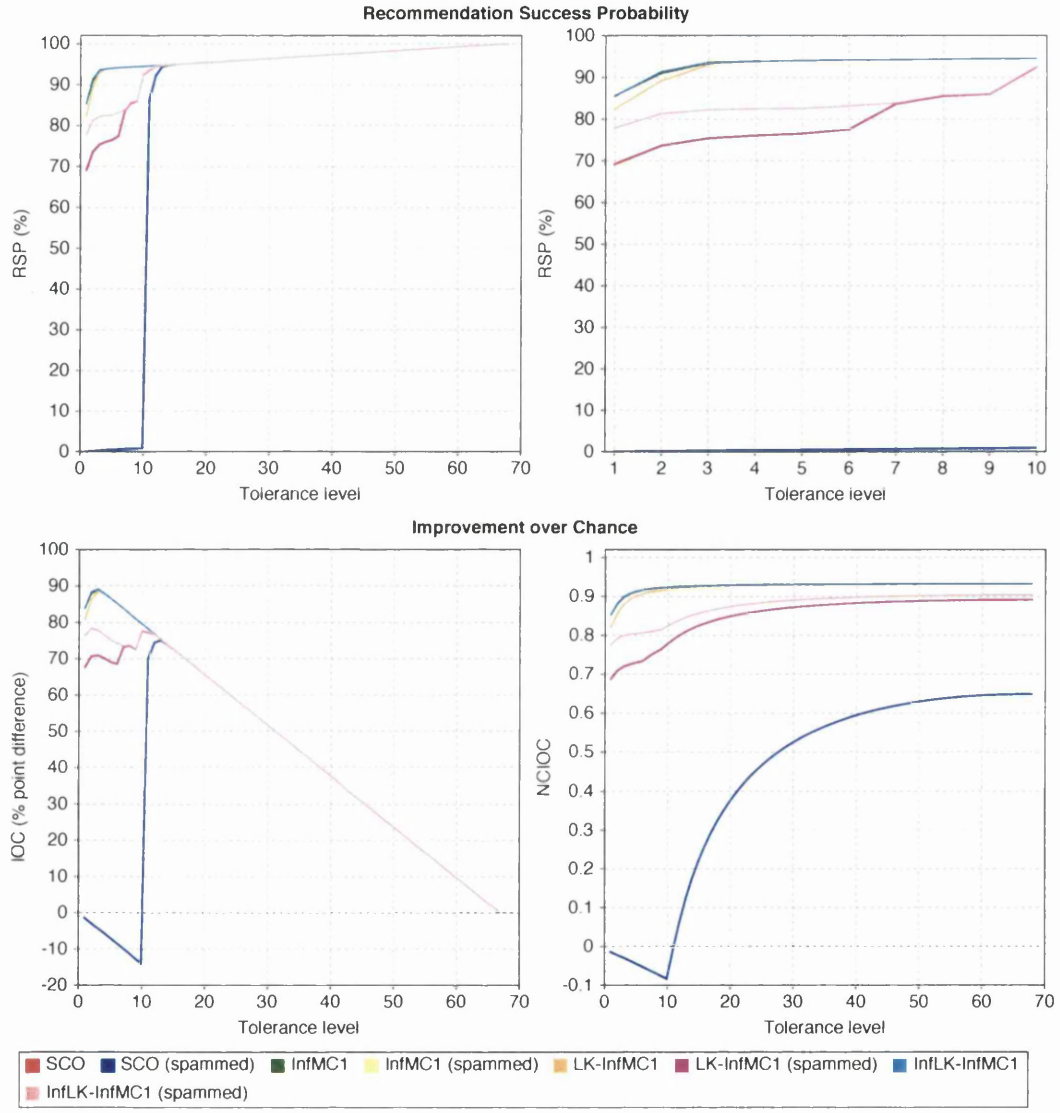
Chapter 10 Details

Recommendation Generation Algorithm	NCIOC (to tolerance level 5)			Best NCIOC	Rank
	Basic	+LK	+InfLK		
SCO	0.3648	N/A	N/A	0.3648	1
Borda	0.1865	0.1908	0.1874	0.1908	6
MC1	0.1981	0.1916	0.1837	0.1981	5
InfMC1	-0.0002	0.2141	0.17	0.2141	4
MC2	0.1794	0.1759	0.1837	0.1837	8
InfMC2	0.1343	0.1302	0.1548	0.1548	10
MC3	0.2441	0.2409	0.1837	0.2441	3
InfMC3	0.1915	0.2857	0.1874	0.2857	2
MC4	0.1297	0.1297	0.1597	0.1597	9
InfMC4	0.142	0.1119	0.1868	0.1868	7

Figure C.1: Experiment One - No Attributes

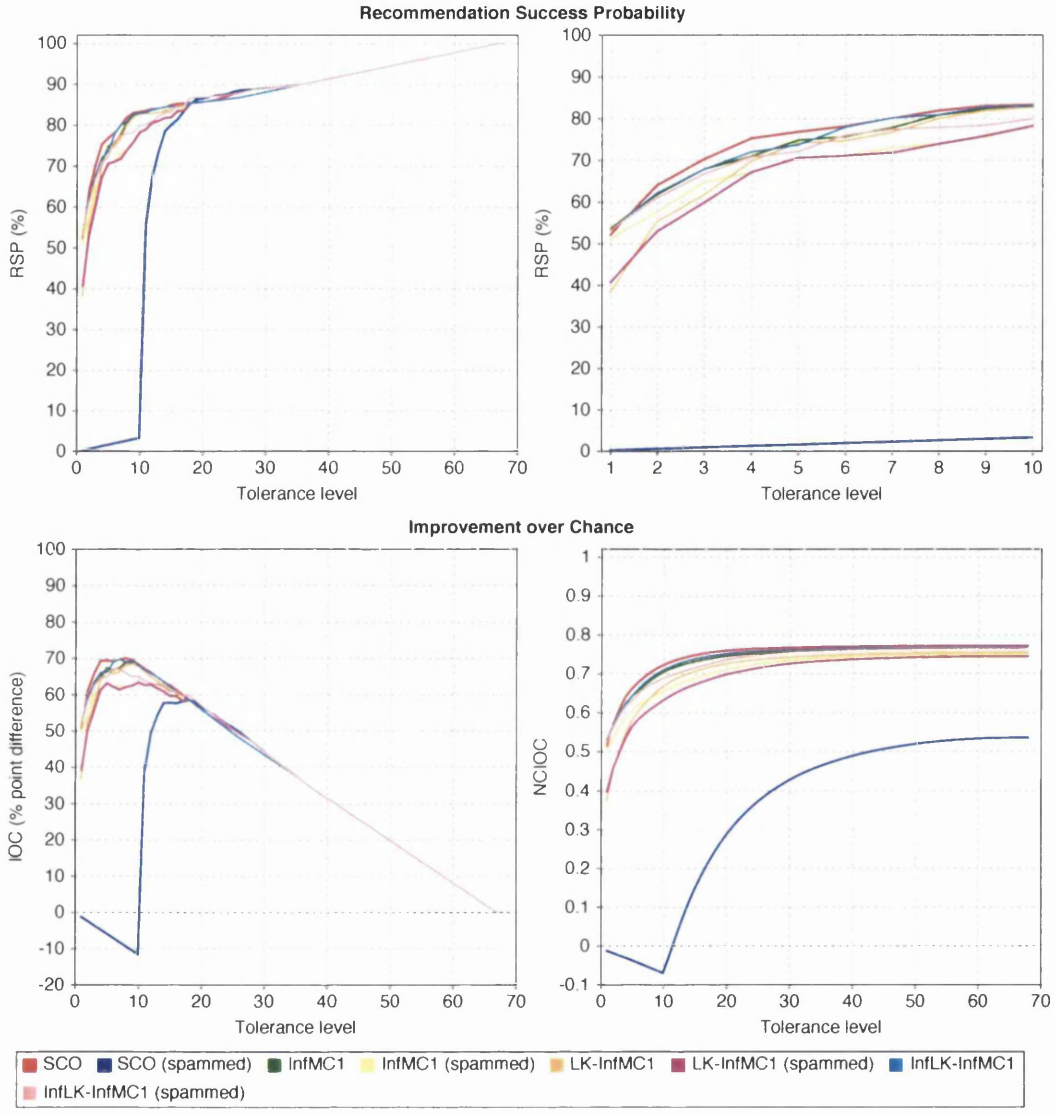
Recommendation Generation Algorithm	NCIOC (to tolerance level 5)			Best NCIOC	Rank
	Basic	+LK	+InfLK		
SCO	0.357	N/A	N/A	0.357	1
Borda	0.325	0.3222	0.3246	0.325	7
MC1	0.1793	0.1797	0.3261	0.3261	4
InfMC1	0.0692	0.1767	0.3284	0.3284	2
MC2	0.2066	0.2072	0.326	0.326	5
InfMC2	0.3124	0.3111	0.3233	0.3233	9
MC3	0.2156	0.2155	0.3257	0.3257	6
InfMC3	0.2943	0.2959	0.3234	0.3234	8
MC4	0.0879	0.0892	0.3224	0.3224	10
InfMC4	0.2122	0.2335	0.328	0.328	3

Figure C.2: Experiment One - HourOfDay



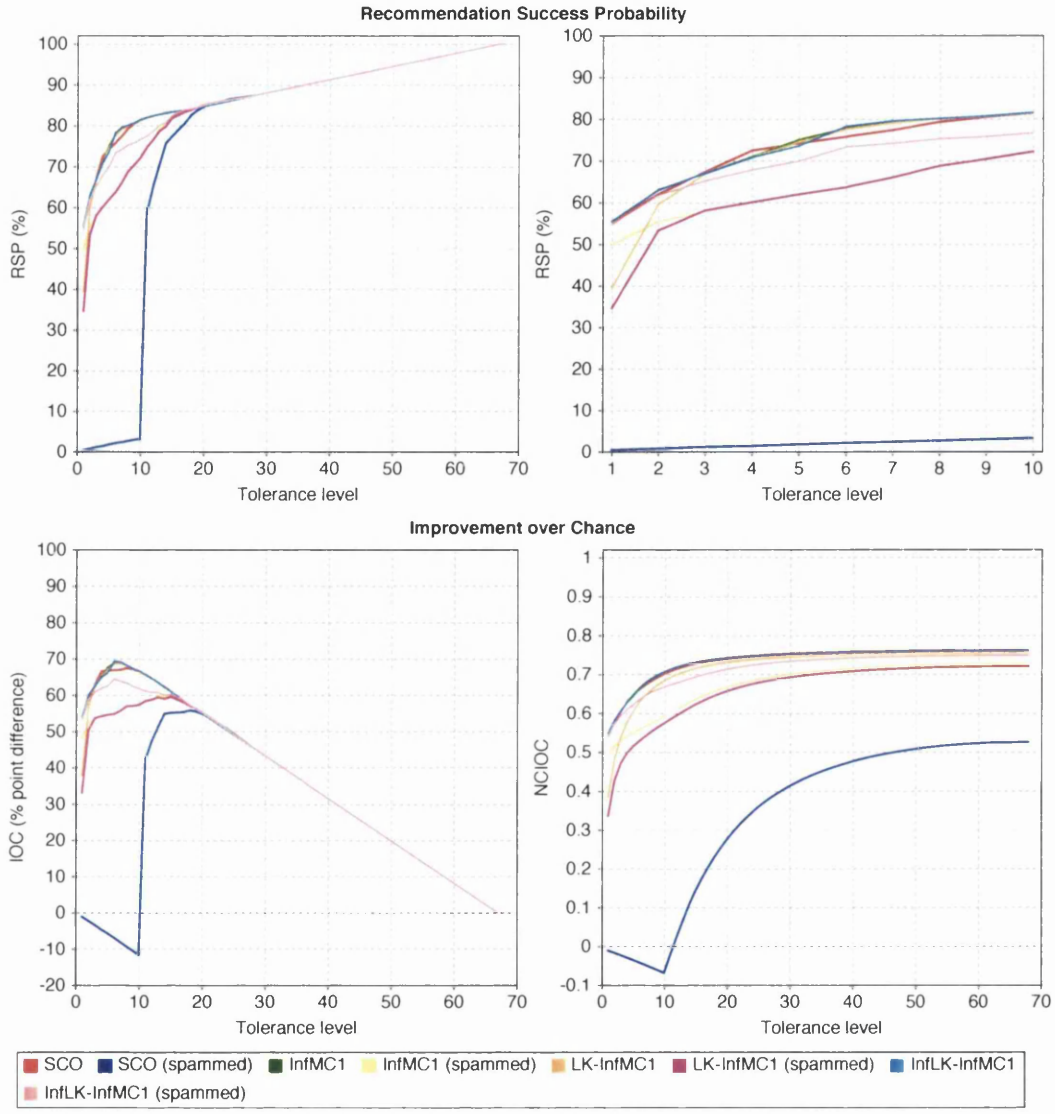
Recommendation Generation Algorithm	Recommendations			Successful Recommendations Info			Cumul. IOC (level 5)	NCIOC (level 5)	Rank
	Success	Failure		Mean # Users	Mean # Services Selected	Mean # Service Selections			
		No service selections	No user situation						
SCO	7824	0	0	64.32	4.1	3522.77	435.82	0.9125	2
SCO (spammed)	7824	0	0	65.32	14.1	36676.46	-21.03	-0.044	8
InfMC1	7824	0	0	64.32	4.1	3522.77	435.49	0.9118	3
InfMC1 (spammed)	7824	0	0	65.32	14.1	36676.46	347.89	0.7284	7
LK-InfMC1	7824	0	0	64.32	4.1	3522.77	430.02	0.9004	4
LK-InfMC1 (spammed)	7824	0	0	65.32	14.1	36676.46	348.46	0.7296	6
InfLK-InfMC1	7824	0	0	64.32	4.1	3522.77	435.99	0.9128	1
InfLK-InfMC1 (spammed)	7824	0	0	65.32	14.1	36676.46	384.21	0.8044	5

Figure C.3: Experiment Two - Location & Role



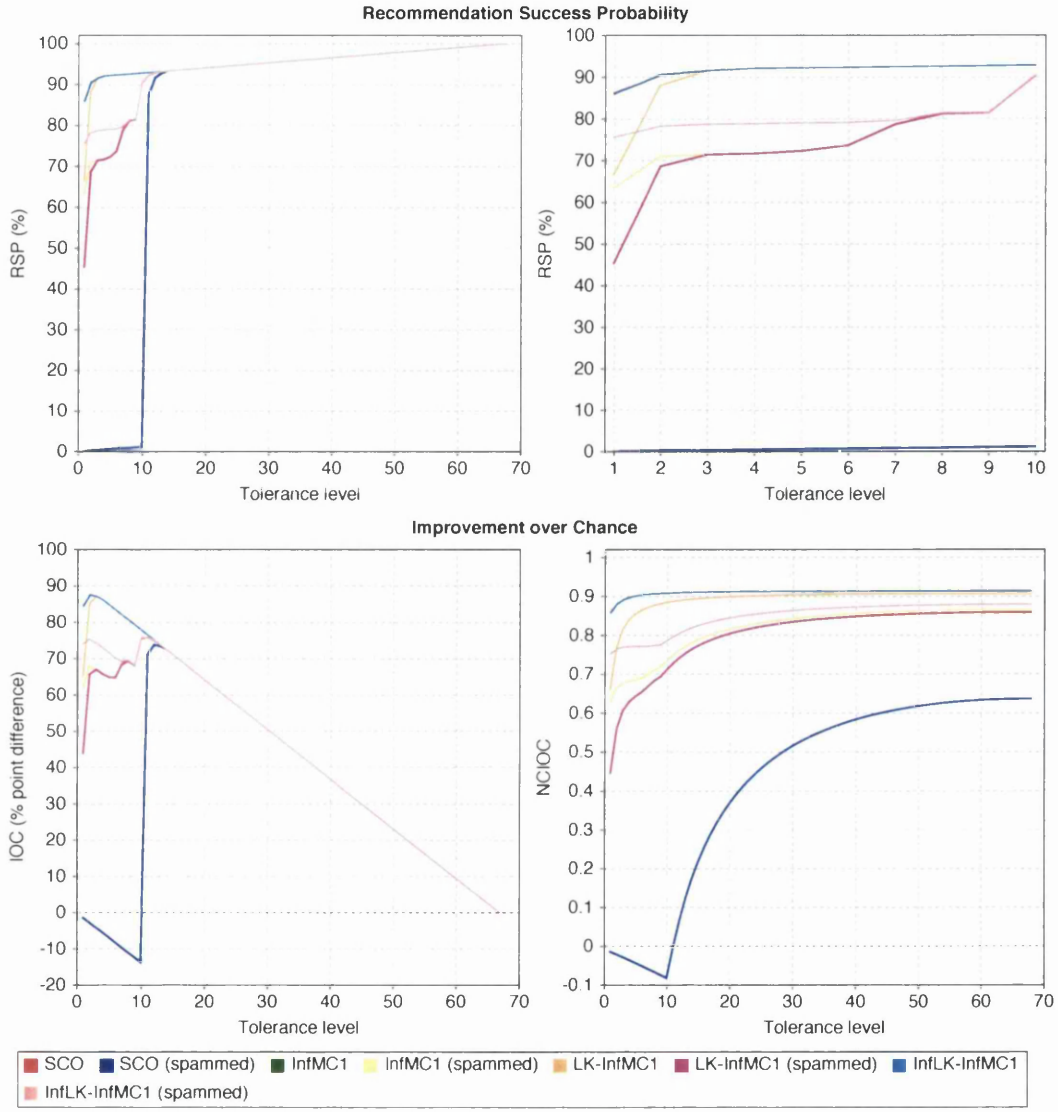
Recommendation Generation Algorithm	Recommendations			Successful Recommendations Info			Cumul. IOC (level 5)	NCIOC (level 5)	Rank
	Success	Failure		Mean # Users	Mean # Services Selected	Mean # Service Selections			
		No service selections	No user situation						
SCO	10360	0	0	75	13.96	4865.22	316.28	0.6622	1
SCO (spammed)	10360	0	0	76	23.96	38643.45	-17.7	-0.0371	8
InfMC1	10360	0	0	75	13.96	4865.22	306.97	0.6427	2
InfMC1 (spammed)	10360	0	0	76	23.96	38643.45	288.83	0.6047	5
LK-InfMC1	10360	0	0	75	13.96	4865.22	276.9	0.5798	6
LK-InfMC1 (spammed)	10360	0	0	76	23.96	38643.45	269.01	0.5632	7
InfLK-InfMC1	10360	0	0	75	13.96	4865.22	306.7	0.6421	3
InfLK-InfMC1 (spammed)	10360	0	0	76	23.96	38643.45	301.57	0.6314	4

Figure C.4: Experiment Two - Role



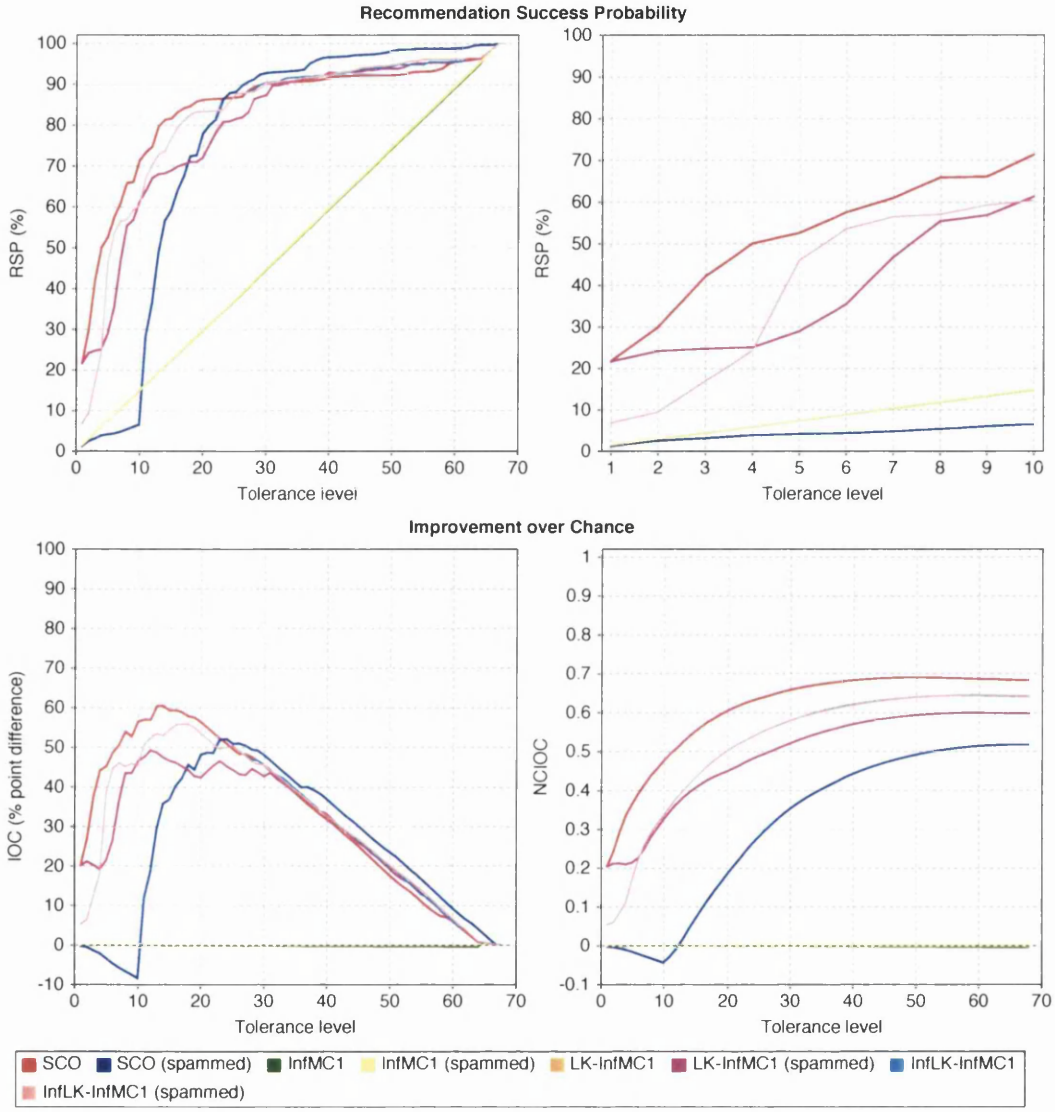
Recommendation Generation Algorithm	Recommendations			Successful Recommendations Info			Cumul. IOC (level 5)	NCIOC (level 5)	Rank
	Success	Failure		Mean # Users	Mean # Services Selected	Mean # Service Selections			
		No service selections	No user situation						
SCO	10268	0	0	51.31	8.32	422.77	309.26	0.6475	2
SCO (spammed)	10268	0	0	52.31	18.32	3544.01	-16.62	-0.0348	8
InfMC1	10268	0	0	51.31	8.32	422.77	309.52	0.648	1
InfMC1 (spammed)	10268	0	0	52.31	18.32	3544.01	263.02	0.5507	6
LK-InfMC1	10268	0	0	51.31	8.32	422.77	288.92	0.6049	5
LK-InfMC1 (spammed)	10268	0	0	52.31	18.32	3544.01	245.88	0.5148	7
InfLK-InfMC1	10268	0	0	51.31	8.32	422.77	307.8	0.6445	3
InfLK-InfMC1 (spammed)	10268	0	0	52.31	18.32	3544.01	297.12	0.6221	4

Figure C.5: Experiment Two - HourOfDay & Role



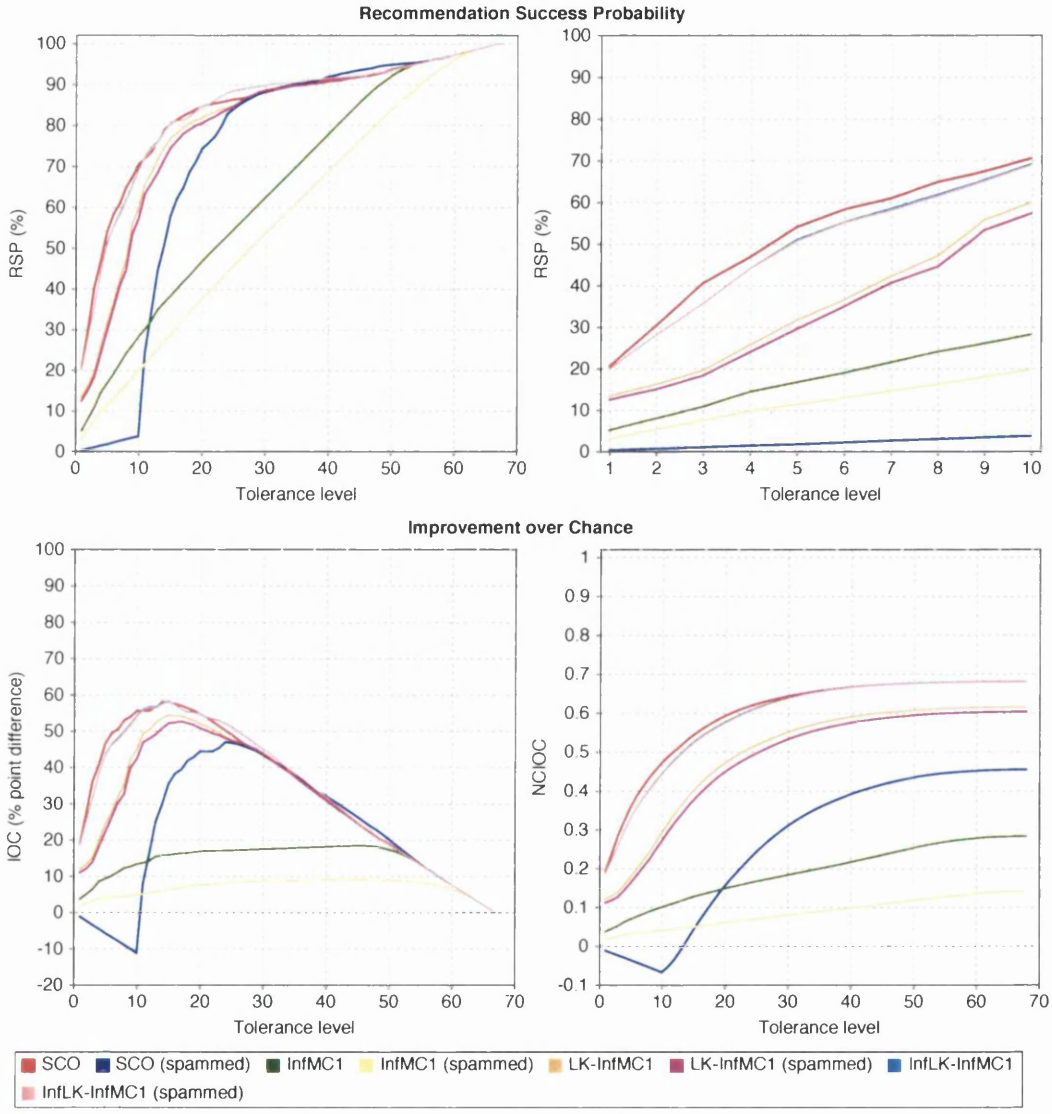
Recommendation Generation Algorithm	Recommendations			Successful Recommendations Info			Cumul. IOC (level 5)	NCIOC (level 5)	Rank
	Success	Failure		Mean # Users	Mean # Services Selected	Mean # Service Selections			
		No service selections	No user situation						
SCO	7288	0	0	46.07	2.62	349.51	430.43	0.9012	1
SCO (spammed)	7288	0	0	47.07	12.62	3698.25	-20.41	-0.0427	8
InfMC1	7288	0	0	46.07	2.62	349.51	430.21	0.9007	3
InfMC1 (spammed)	7288	0	0	47.07	12.62	3698.25	327.73	0.6862	6
LK-InfMC1	7288	0	0	46.07	2.62	349.51	408.43	0.8551	4
LK-InfMC1 (spammed)	7288	0	0	47.07	12.62	3698.25	307.35	0.6435	7
InfLK-InfMC1	7288	0	0	46.07	2.62	349.51	430.37	0.9011	2
InfLK-InfMC1 (spammed)	7288	0	0	47.07	12.62	3698.25	368.48	0.7715	5

Figure C.6: Experiment Two - HourOfDay & Location & Role



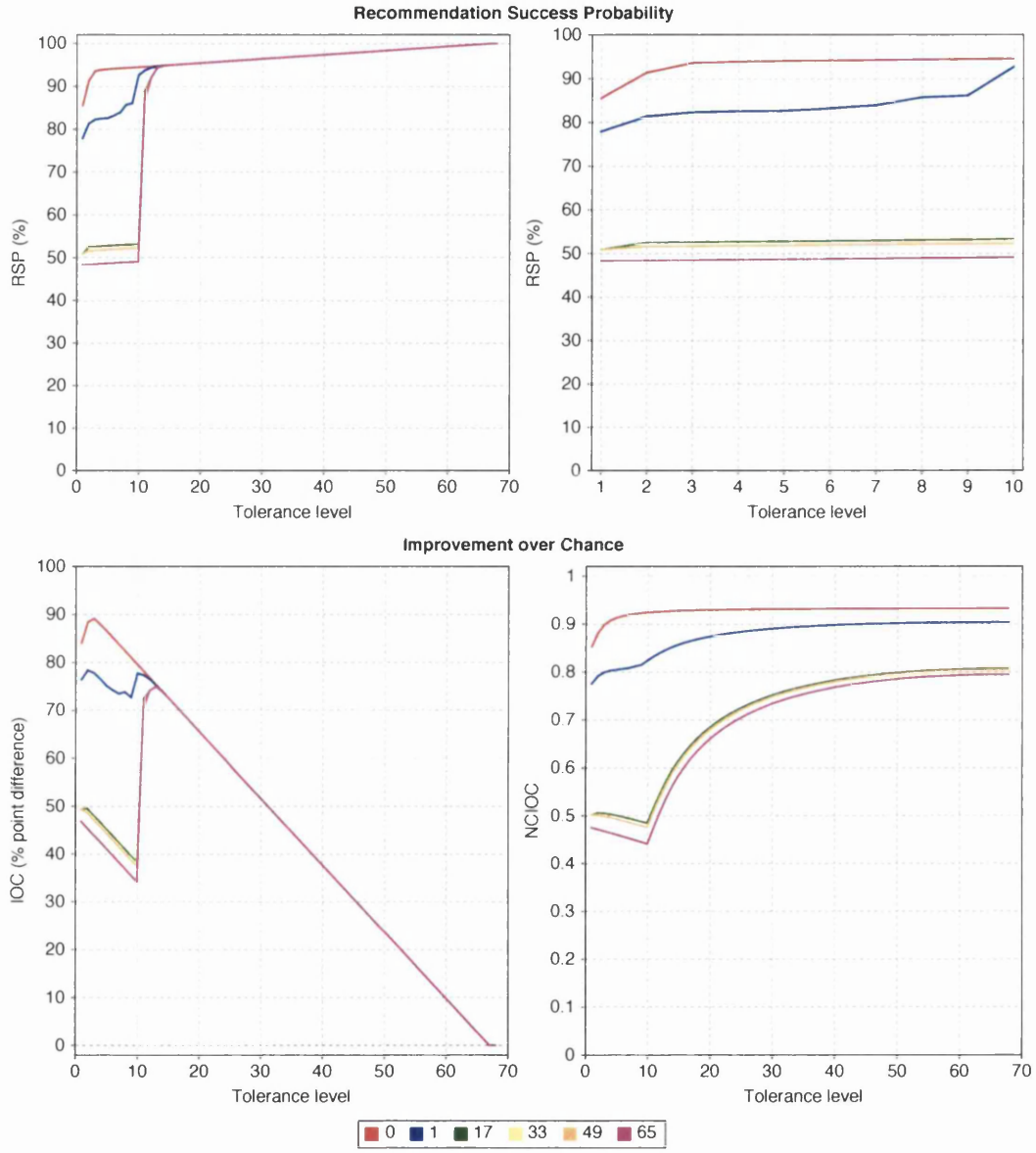
Recommendation Generation Algorithm	Recommendations			Successful Recommendations Info			Cumul. IOC (level 5)	NCIOC (level 5)	Rank
	Success	Failure		Mean # Users	Mean # Services Selected	Mean # Service Selections			
		No service selections	No user situation						
SCO	10397	0	0	868.54	64.96	39342.84	174.23	0.3648	1
SCO (spammed)	10397	0	0	869.54	67.02	127193.65	-7.49	-0.0157	6
InfMC1	10397	0	0	868.54	64.96	39342.84	-0.11	-0.0002	5
InfMC1 (spammed)	10397	0	0	869.54	67.02	127193.65	0	0	4
LK-InfMC1	10397	0	0	868.54	64.96	39342.84	102.26	0.2141	2
LK-InfMC1 (spammed)	10397	0	0	869.54	67.02	127193.65	102.26	0.2141	2
InfLK-InfMC1	10397	0	0	868.54	64.96	39342.84	81.18	0.17	3
InfLK-InfMC1 (spammed)	10397	0	0	869.54	67.02	127193.65	81.18	0.17	3

Figure C.7: Experiment Two - No Attributes



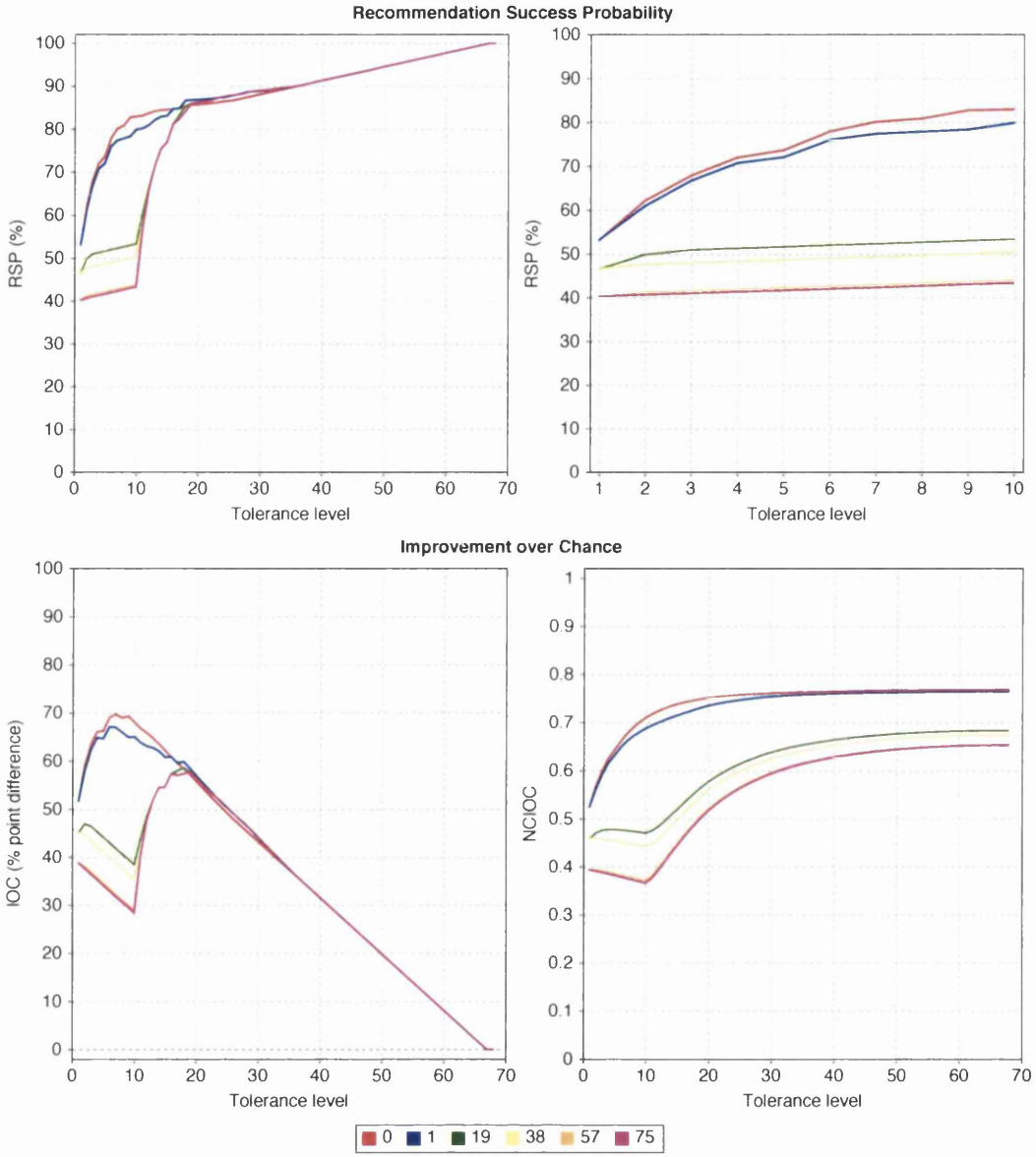
Recommendation Generation Algorithm	Recommendations			Successful Recommendations Info			Cumul. IOC (level 5)	NCIOC (level 5)	Rank
	Success	Failure		Mean # Users	Mean # Services Selected	Mean # Service Selections			
		No service selections	No user situation						
SCO	10397	0	0	460.67	45.83	3257.31	170.49	0.357	1
SCO (spammed)	10397	0	0	461.67	55.83	10823.63	-17	-0.0356	8
InfMC1	10397	0	0	460.67	45.83	3257.31	33.06	0.0692	6
InfMC1 (spammed)	10397	0	0	461.67	55.83	10823.63	14.81	0.031	7
LK-InfMC1	10397	0	0	460.67	45.83	3257.31	84.41	0.1767	4
LK-InfMC1 (spammed)	10397	0	0	461.67	55.83	10823.63	77.23	0.1617	5
InfLK-InfMC1	10397	0	0	460.67	45.83	3257.31	156.84	0.3284	2
InfLK-InfMC1 (spammed)	10397	0	0	461.67	55.83	10823.63	156.41	0.3275	3

Figure C.8: Experiment Two - HourOfDay



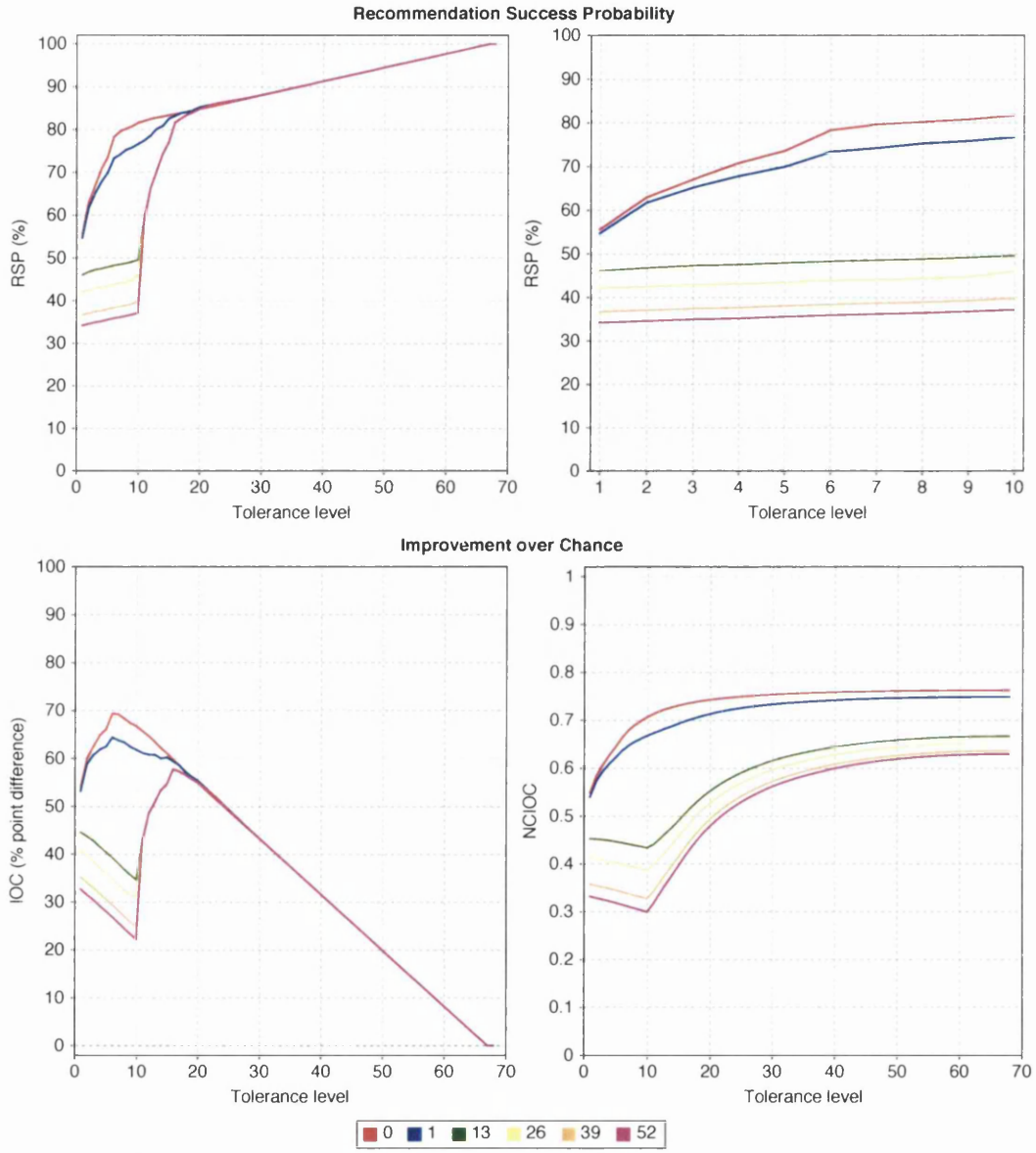
# Spamming Voters	Recommendations			Successful Recommendations Info			Cumul. IOC (level 5)	NCIOC (level 5)	Rank
	Success	Failure		Mean # Users	Mean # Services Selected	Mean # Service Selections			
		No service selections	No user situation						
0	7824	0	0	64.32	4.1	3522.77	435.99	0.9128	1
1	7824	0	0	65.32	14.1	36676.46	384.21	0.8044	2
17	7824	0	0	81.32	14.1	567135.48	238.98	0.5004	3
33	7824	0	0	97.32	14.1	1097594.5	235.3	0.4927	4
49	7824	0	0	113.32	14.1	1628053.52	235.3	0.4927	4
65	7824	0	0	129.32	14.1	2158512.55	219.83	0.4603	5

Figure C.9: Experiment Three - Location & Role



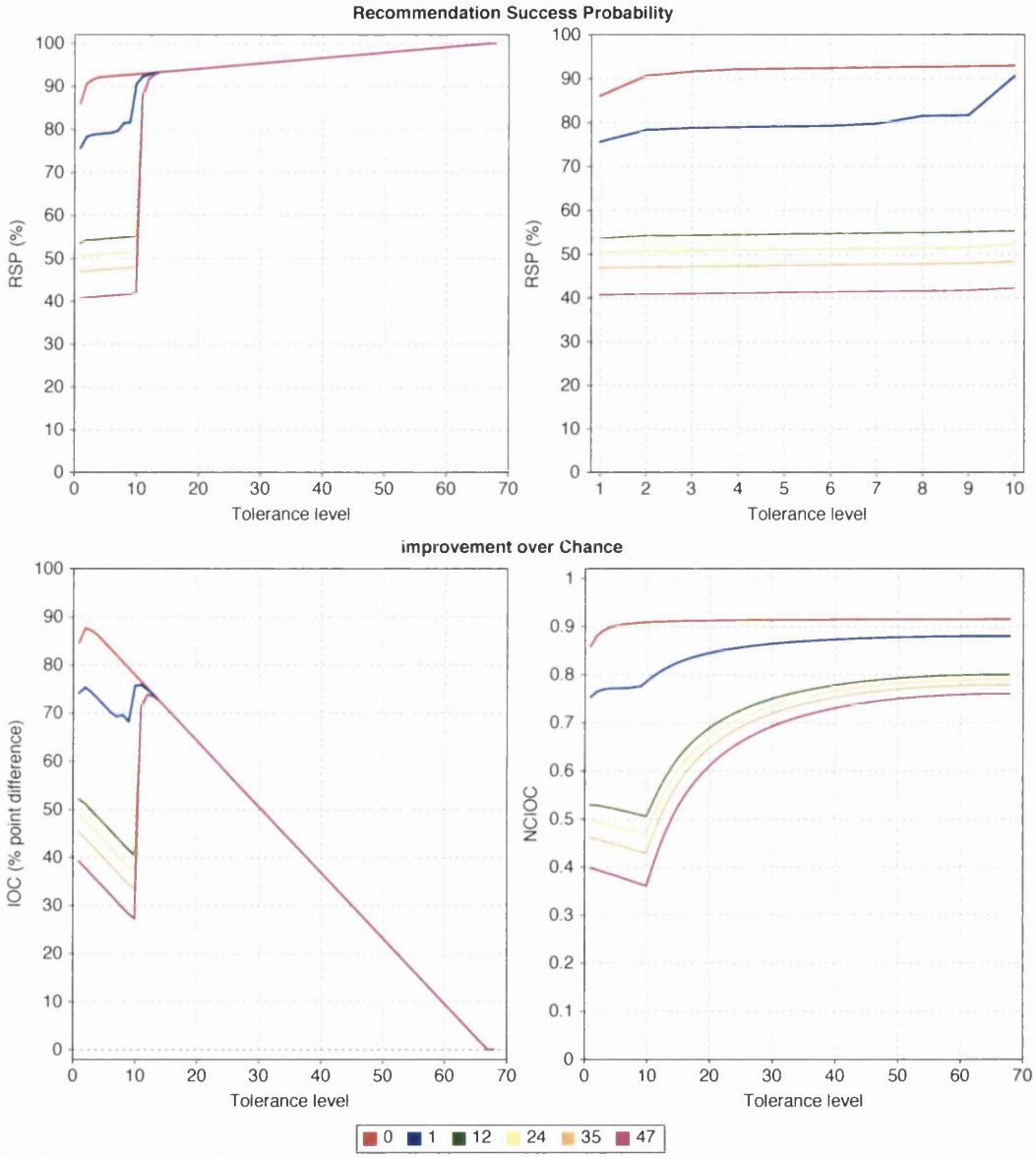
# Spamming Voters	Recommendations			Successful Recommendations Info			Cumul. IOC (level 5)	NCIOC (level 5)	Rank
	Success	Failure		Mean # Users	Mean # Services Selected	Mean # Service Selections			
		No service selections	No user situation						
0	10360	0	0	75	13.96	4865.22	306.7	0.6421	1
1	10360	0	0	76	23.96	38643.45	301.57	0.6314	2
19	10360	0	0	94	23.96	646651.64	228.07	0.4775	3
38	10360	0	0	113	23.96	1288438.06	216.87	0.4541	4
57	10360	0	0	132	23.96	1930224.48	184.9	0.3871	5
75	10360	0	0	150	23.96	2538232.66	182.54	0.3822	6

Figure C.10: Experiment Three - Role



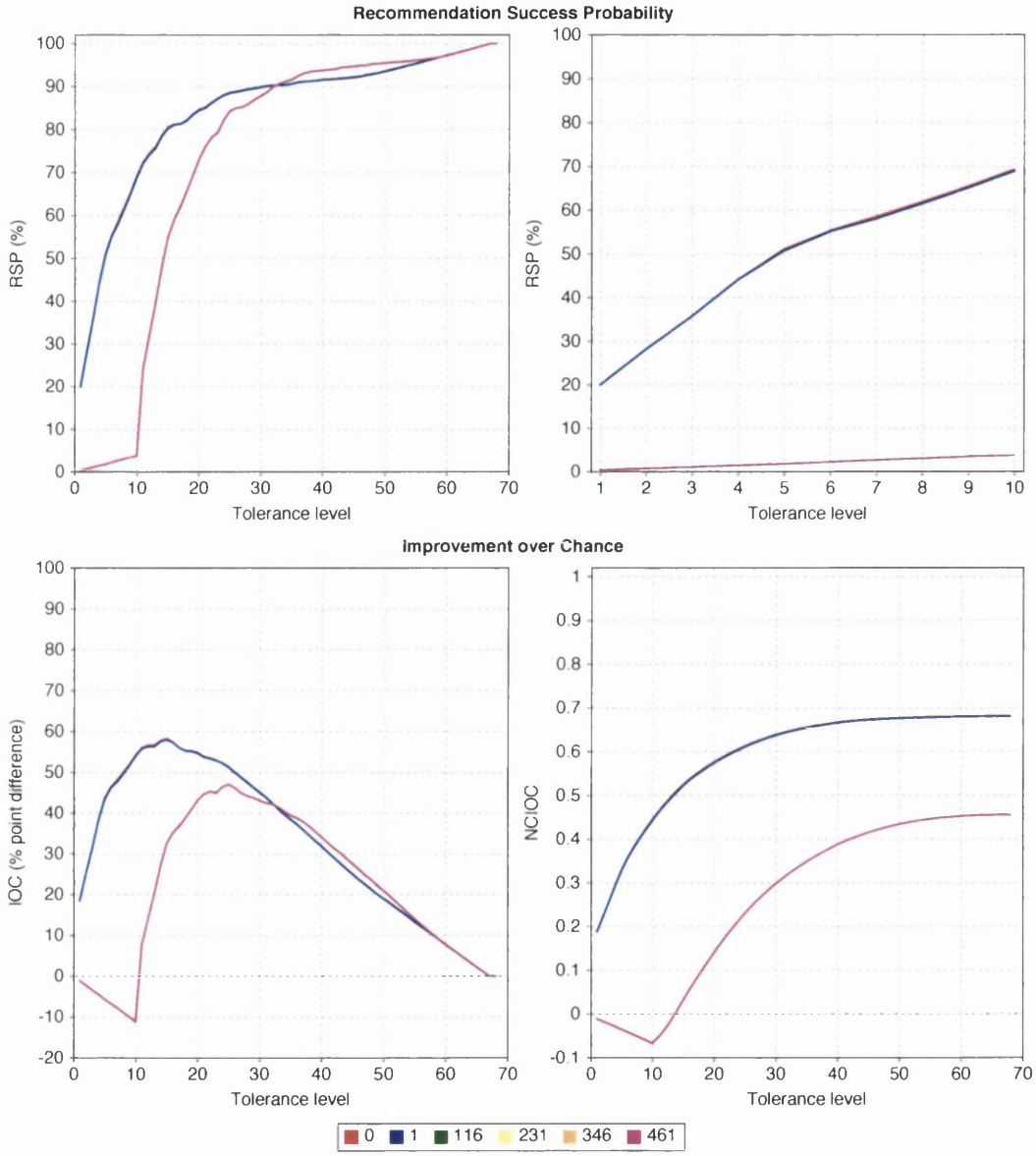
# Spamming Voters	Recommendations			Successful Recommendations Info			Cumul. IOC (level 5)	NCIOC (level 5)	Rank
	Success	Failure		Mean # Users	Mean # Services Selected	Mean # Service Selections			
		No service selections	No user situation						
0	10268	0	0	51.31	8.32	422.77	307.8	0.6445	1
1	10268	0	0	52.31	18.32	3544.01	297.12	0.6221	2
13	10268	0	0	64.31	18.32	40998.93	213.25	0.4465	3
26	10268	0	0	77.31	18.32	81575.09	191.94	0.4019	4
39	10268	0	0	90.31	18.32	122151.25	164.72	0.3449	5
52	10268	0	0	103.31	18.32	162727.41	152.11	0.3185	6

Figure C.11: Experiment Three - HourOfDay & Role



# Spamming Voters	Recommendations			Successful Recommendations Info			Cumul. IOC (level 5)	NCIOC (level 5)	Rank
	Success	Failure		Mean # Users	Mean # Services Selected	Mean # Service Selections			
		No service selections	No user situation						
0	7288	0	0	46.07	2.62	349.51	430.37	0.9011	1
1	7288	0	0	47.07	12.62	3698.25	368.48	0.7715	2
12	7288	0	0	58.07	12.62	40534.37	248.76	0.5208	3
24	7288	0	0	70.07	12.62	80719.23	231.64	0.485	4
35	7288	0	0	81.07	12.62	117555.35	213.54	0.4471	5
47	7288	0	0	93.07	12.62	157740.21	182.46	0.382	6

Figure C.12: Experiment Three - HourOfDay & Location & Role

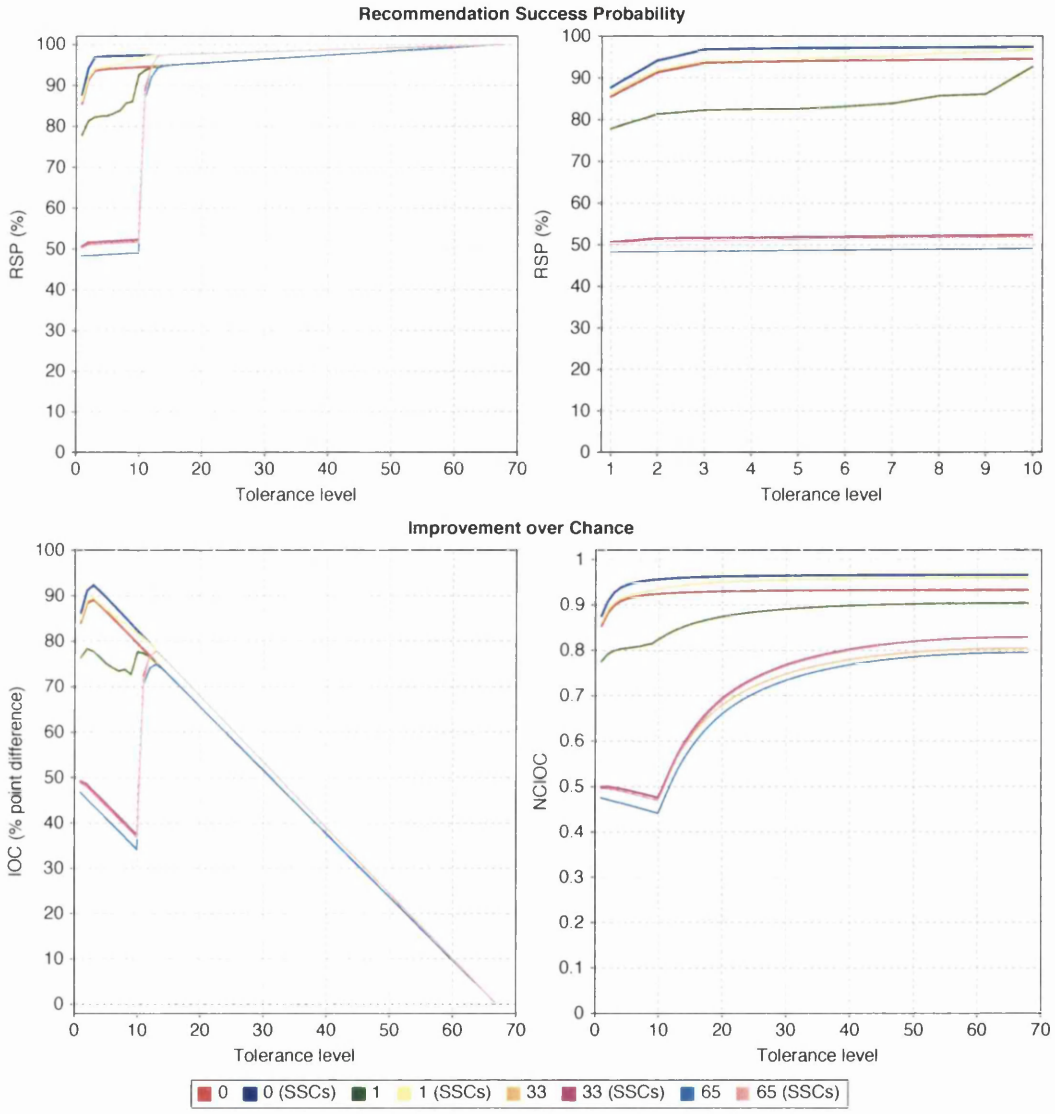


# Spamming Voters	Recommendations			Successful Recommendations Info			Cumul. IOC (level 5)	NCIOC (level 5)	Rank
	Success	Failure		Mean # Users	Mean # Services Selected	Mean # Service Selections			
		No service selections	No user situation						
0	10397	0	0	460.67	45.83	3257.31	156.84	0.3284	1
1	10397	0	0	461.67	55.83	10823.63	156.41	0.3275	2
116	10397	0	0	576.67	55.83	880949.45	-17	-0.0356	3
231	10397	0	0	691.67	55.83	1751075.28	-17	-0.0356	3
346	10397	0	0	806.67	55.83	2621201.1	-17	-0.0356	3
461	10397	0	0	921.67	55.83	3491326.93	-17	-0.0356	3

Figure C.13: Experiment Three - HourOfDay

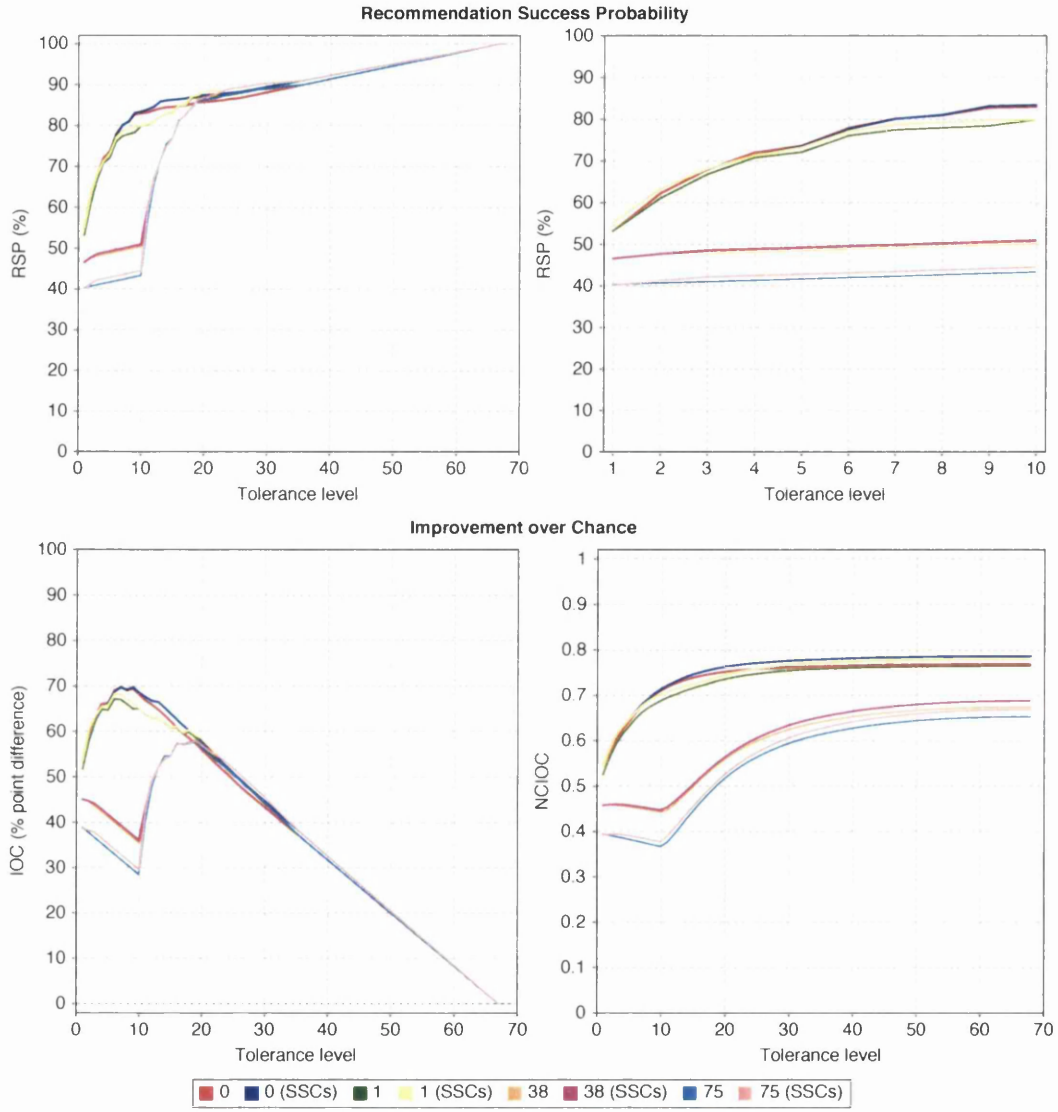
Appendix D

Chapter 11 Details



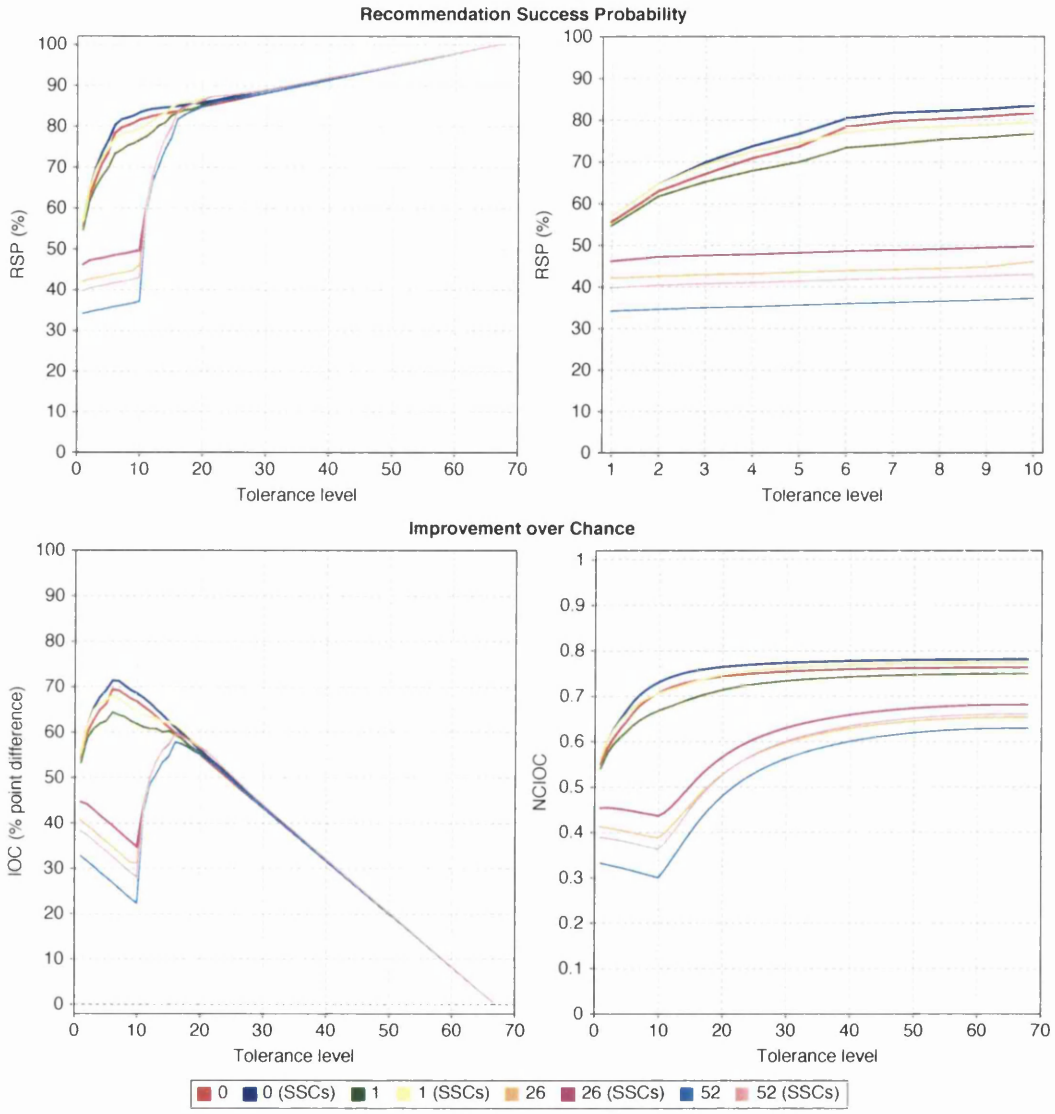
# Spamming Individuals	Recommendations			Successful Recommendations Info			Cumul. IOC (level 5)	NCIOC (level 5)	Rank
	Success	Failure		Mean # Users	Mean # Services Selected	Mean # Service Selections			
		No service selections	No user situation						
0	7824	0	0	64.32	4.1	3522.77	435.99	0.9128	3
0 (SSCs)	7824	0	0	79.15	5.21	4536.24	450.53	0.9433	1
1	7824	0	0	65.32	14.1	36676.46	384.21	0.8044	4
1 (SSCs)	7824	0	0	80.15	15.03	37689.92	438.79	0.9187	2
33	7824	0	0	97.32	14.1	1097594.5	235.3	0.4927	6
33 (SSCs)	7824	0	0	112.15	15.03	1098607.97	235.36	0.4928	5
65	7824	0	0	129.32	14.1	2158512.55	219.83	0.4603	8
65 (SSCs)	7824	0	0	144.15	15.03	2159526.01	232.41	0.4866	7

Figure D.1: Experiment Two - Location & Role



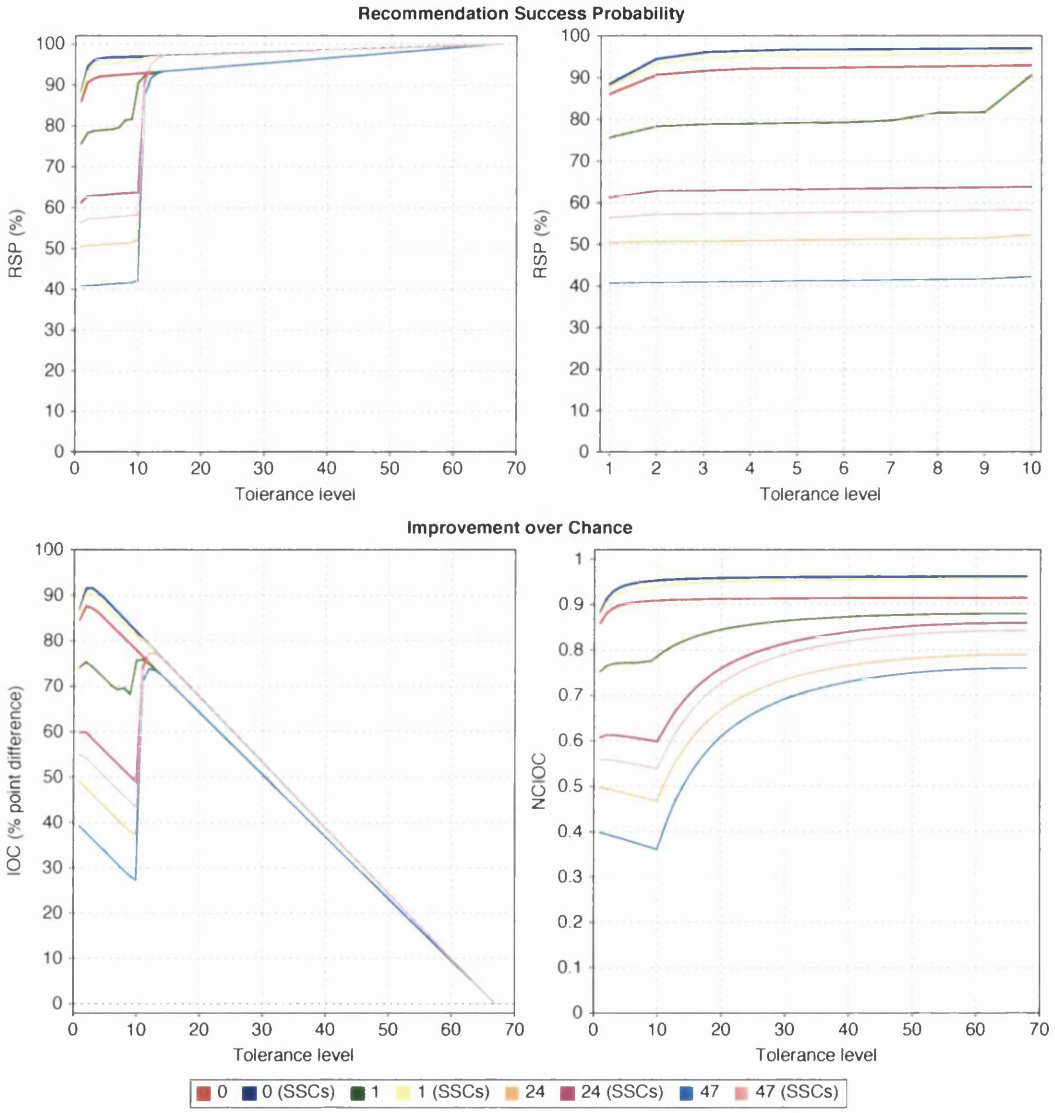
# Spamming Individuals	Recommendations			Successful Recommendations Info			Cumul. IOC (level 5)	NCIOC (level 5)	Rank
	Success	Failure		Mean # Users	Mean # Services Selected	Mean # Service Selections			
		No service selections	No user situation						
0	10360	0	0	75	13.96	4865.22	306.7	0.6421	3
0 (SSCs)	10360	0	0	89.39	14.87	5641.07	309.07	0.6471	1
1	10360	0	0	76	23.96	38643.45	301.57	0.6314	4
1 (SSCs)	10360	0	0	90.39	24.72	39419.3	308.86	0.6467	2
38	10360	0	0	113	23.96	1288438.06	216.87	0.4541	6
38 (SSCs)	10360	0	0	127.39	24.72	1289213.91	218.67	0.4578	5
75	10360	0	0	150	23.96	2538232.66	182.54	0.3822	8
75 (SSCs)	10360	0	0	164.39	24.72	2539008.52	186.54	0.3906	7

Figure D.2: Experiment Two - Role



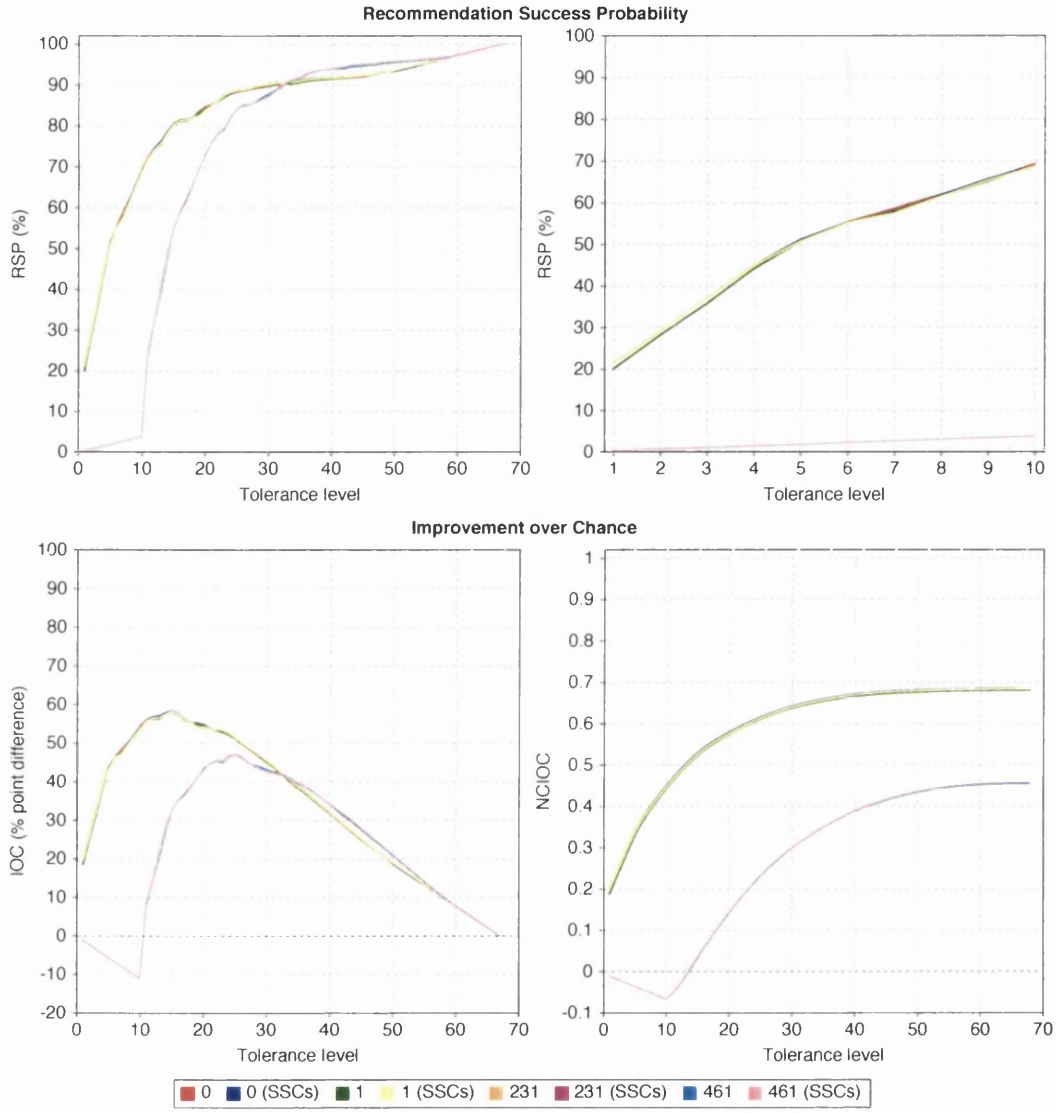
# Spamming Individuals	Recommendations			Successful Recommendations Info			Cumul. IOC (level 5)	NCIOC (level 5)	Rank
	Success	Failure		Mean # Users	Mean # Services Selected	Mean # Service Selections			
		No service selections	No user situation						
0	10268	0	0	51.31	8.32	422.77	307.8	0.6445	3
0 (SSCs)	10268	0	0	71.42	9.65	1636.62	319.55	0.6691	1
1	10268	0	0	52.31	18.32	3544.01	297.12	0.6221	4
1 (SSCs)	10268	0	0	72.42	19.42	4757.86	315.51	0.6606	2
26	10268	0	0	77.31	18.32	81575.09	191.94	0.4019	6
26 (SSCs)	10268	0	0	97.42	19.42	82788.94	214.48	0.4491	5
52	10268	0	0	103.31	18.32	162727.41	152.11	0.3185	8
52 (SSCs)	10268	0	0	123.42	19.42	163941.26	180.94	0.3788	7

Figure D.3: Experiment Two - HourOfDay & Role



# Spamming Individuals	Recommendations			Successful Recommendations Info			Cumul. IOC (level 5)	NCIOC (level 5)	Rank
	Success	Failure		Mean # Users	Mean # Services Selected	Mean # Service Selections			
		No service selections	No user situation						
0	7288	0	0	46.07	2.62	349.51	430.37	0.9011	3
0 (SSCs)	7288	0	0	80.93	4.74	2892	449.89	0.9419	1
1	7288	0	0	47.07	12.62	3698.25	368.48	0.7715	4
1 (SSCs)	7288	0	0	81.93	14.42	6240.74	443.86	0.9293	2
24	7288	0	0	70.07	12.62	80719.23	231.64	0.485	7
24 (SSCs)	7288	0	0	104.93	14.42	83261.71	290.91	0.6091	5
47	7288	0	0	93.07	12.62	157740.21	182.46	0.382	8
47 (SSCs)	7288	0	0	127.93	14.42	160282.69	263.69	0.5521	6

Figure D.4: Experiment Two - HourOfDay & Location & Role



# Spamming Individuals	Recommendations			Successful Recommendations Info			Cumul. IOC (level 5)	NCIOC (level 5)	Rank
	Success	Failure		Mean # Users	Mean # Services Selected	Mean # Service Selections			
		No service selections	No user situation						
0	10397	0	0	460.67	45.83	3257.31	156.84	0.3284	3
0 (SSCs)	10397	0	0	499.42	46.58	4104.44	161.19	0.3375	1
1	10397	0	0	461.67	55.83	10823.63	156.41	0.3275	4
1 (SSCs)	10397	0	0	500.42	56.29	11670.75	160.85	0.3368	2
231	10397	0	0	691.67	55.83	1751075.28	-17	-0.0356	5
231 (SSCs)	10397	0	0	730.42	56.29	1751922.4	-17	-0.0356	5
461	10397	0	0	921.67	55.83	3491326.93	-17	-0.0356	5
461 (SSCs)	10397	0	0	960.42	56.29	3492174.05	-17	-0.0356	5

Figure D.5: Experiment Two - HourOfDay

References

[1] William Adjie-Winoto, Elliot Schwartz, Hari Balakrishnan, and Jeremy Lilley. The Design and Implementation of an Intentional Naming System. *Operating Systems Review*, 34(5), 1999.

[2] Rama Akkiraju, Richard Goodwin, Prashant Doshi, and Sascha Roeder. A Method for Semantically Enhancing the Service Discovery Capabilities of UDDI. In *IJCAI-03 Workshop on Information Integration on the Web*, 2003.

[3] Amazon.com. Amazon Book Recommender. http://www.amazon.com. Accessed July 2004.

[4] Javed A. Aslam and Mark Montague. Models for Metasearch. In *24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, 2001.

[5] Wolf-Tilo Balke and Matthias Wagner. Towards Personalised Selection of Web Services. In *12th International World Wide Web Conference (WWW)*, 2003.

[6] Tim Berners-Lee, James Hendler, and Ora Lassila. The Semantic Web. *Scientific American*, 284(5), May 2001.

[7] Christian Bettstetter and Christoph Renner. A Comparison of Service Discovery Protocols and Implementation of the Service Location Protocol. In *EUNICE Open European Summer School*, 2000.

[8] Guy Bieber and Jeff Carpenter. Introduction to Service-Oriented Programming (Rev 2.1). Openwings whitepaper. http://www.openwings.org/download/specs/ServiceOrientedIntroduction.pdf. Accessed April 2004.

[9] Gunter Bolch, Stefan Greiner, Hermann de Meer, and Kishor S. Trivedi. *Queueing Networks and Markov Chains: Modelling and Performance Evaluation with Computer Science Applications*, chapter 2: Markov Chains. John Wiley and Sons, Inc, 1998.

[10] Gaetano Borriello and Roy Want. Embedded Computation Meets the World Wide Web. *Communications of the ACM*, 43(5), 2000.

[11] Mauro Brunato and Roberto Battiti. A Location-Dependent Recommender System for the Web. In *MobEA Workshop*, 2003.

[12] Mauro Brunato and Roberto Battiti. PILGRIM: A Location Broker and Mobility-Aware Recommendation System. In *1st IEEE International Conference on Pervasive Computing and Communications (PerCom)*, 2003.

[13] Rajkumar Buyya and Srikumar Venugopal. The Gridbus Toolkit for Service Oriented Grid and Utility Computing: An Overview and Status Report. In *1st IEEE International Workshop on Grid Economics and Business Models*, 2004.

[14] Matthew Chalmers. Paths and Contextually Specific Recommendations. In *2nd DELOS Network of Excellence Workshop on Personalisation and Recommender Systems in Digital Libraries*, 2001.

[15] Matthew Chalmers, Kerry Rodden, and Dominique Brodbeck. The Order of Things: Activity-Centred Information Access. In *7th International World Wide Web Conference (WWW)*, 1998.

[16] Harry Chen, Anupam Joshi, and Timothy W. Finin. Dynamic Service Discovery for Mobile Computing: Intelligent Agents Meet Jini in the Aether. *Cluster Computing*, 4(4), 2001.

[17] The DAML-S Coalition. DAML-S: Web Service Description for the Semantic Web. In *International Semantic Web Conference (ISWC)*, 2002.

[18] The OWL Services Coalition. OWL-S: Semantic Markup for Web Services. Technical whitepaper. http://www.daml.org/services/owl-s/1.0/owl-s.pdf. Accessed May 2004.

[19] Salutation Consortium. Salutation Architecture: Overview. Technical whitepaper. http://www.salutation.org/whitepaper/originalwp.pdf. Accessed May 2004.

[20] World Wide Web Consortium. Extensible Markup Language (XML) Version 1.1. http://www.w3.org/TR/2004/REC-xml11-20040204. Accessed July 2004.

[21] World Wide Web Consortium. SOAP Version 1.2. http://www.w3.org/TR/2003/REC-soap12-part1-20030624. Accessed April 2004.

[22] World Wide Web Consortium. Web Services Activity. http://www.w3.org/2002/ws. Accessed April 2004.

[23] World Wide Web Consortium. Web Services Description Language (WSDL) Version 1.1. http://www.w3.org/TR/wsdl. Accessed April 2004.

[24] William S. Cooper. Expected Search Length: A Single Measure of Retrieval Effectiveness Based on the Weak Ordering Action of Retrieval Systems. *Journal of the American Society for Information Science*, 19, 1968.

[25] Patricia Dockhorn Costa, José Gonçalves Pereira Filho, and Marten van Sinderen. Architectural Requirements for Building Context-Aware Service Platforms. In *9th EUNICE Open European Summer School and IFIP Workshop on Next Generation Networks*, 2003.

[26] Francisco Curbera, Matthew Duftler, Rania Khalaf, William Nagy, Nirmal Mukhi, and Sanjiva Weerawarana. Unraveling the Web Services Web. *IEEE Internet Computing*, 6(2), 2002.

[27] Steven E. Czerwinski, Ben Y. Zhao, Todd D. Hodes, Anthony D. Joseph, and Randy H. Katz. An Architecture for a Secure Service Discovery Service. In *5th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, 1999.

[28] Todd Datz. What You Need to Know About Service-Oriented Architecture. http://www.cio.com/archive/011504/soa.html. Accessed April 2004.

[29] Jean-Charles de Borda. Mémoire sur les Élections au Scrutin. Histoire de l'Académie Royale des Sciences, 1781.

[30] Anind K. Dey, Gregory D. Abowd, and Daniel Salber. A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. *Human-Computer Interaction Journal, Special Issue on Context-Aware Computing*, 16(1), 2001.

[31] Vijay Dheap, Mohammed A. Munawar, and Paul A. S. Ward. Service-Based Pervasive Computing Environment. Submitted to 19th ACM Symposium on Applied Computing (SAC), Ubiquitous Computing Track, 2004.

[32] Persi Diaconis and R. L. Graham. Spearman's Footrule as a Measure of Disarray. *Journal of the Royal Statistical Society, Series B (Methodological)*, 39(2), 1977.

[33] Remco Dijkman, Dick Quartel, Luis Ferreira Pires, and Marten van Sinderen. The State of the Art in Service-Oriented Computing and Design. Technical Report ArCo/WP1/T1/V1.00, University of Twente, 2003.

[34] Mark Dunlop. Reflections on Mira: Interactive Evaluation in Information Retrieval. *Journal of American Society for Information Science*, 51(14), 2000.

[35] Cynthia Dwork, Ravi Kumar, Moni Naor, and D. Sivakumar. Rank Aggregation Methods for the Web. In *10th International World Wide Web Conference (WWW)*, 2001.

[36] Cynthia Dwork, Ravi Kumar, Moni Naor, and D. Sivakumar. Rank Aggregation Revisited. Manuscript, 2001.

[37] W. Keith Edwards. *Core Jini.* Prentice Hall, 1999.

[38] W. Keith Edwards, Mark W. Newman, Jana Sedivy, Trevor Smith, and Shahram Izadi. Challenge: Recombinant Computing and the Speakeasy Approach. In *8th Annual International Conference on Mobile Computing and Networking (MobiCom)*, 2002.

[39] Ronald Fagin, Ravi Kumar, Mohammad Mahdian, D. Sivakumar, and Erik Vee. Comparing and Aggregating Rankings with Ties. In *2004 ACM Symposium on Principles of Database Systems (PODS)*, 2004.

[40] Ian Foster and Carl Kesselman (editors). *The Grid: Blueprint for a Future Computing Infrastructure.* Morgan Kaufmann, 1999.

[41] Ian Foster, Carl Kesselman, Jeffrey M. Nick, and Steven Tuecke. Grid Services for Distributed System Integration. *Computer*, 35(6), 2002.

[42] Ian Foster, Carl Kesselman, Jeffrey M. Nick, and Steven Tuecke. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration.

Technical report, Open Grid Service Infrastructure Working Group, Global Grid Forum, 2002.

[43] Ian Foster, Carl Kesselman, and Steven Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organisations. *International J. Supercomputer Applications*, 15(3), 2001.

[44] Adrian Friday, Nigel Davies, Nat Wallbank, Elaine Catterall, and Stephen Pink. Supporting Service Discovery, Querying and Interaction in Ubiquitous Computing Environments. *ACM Baltzer Wireless Networks (WINET) Special Issue*, 10(6), 2004.

[45] Globus. The Globus Toolkit. http://www-unix.globus.org/toolkit. Accessed May 2004.

[46] Ken Goldberg, Theresa Roeder, Dhruv Gupta, and Chris Perkins. Eigentaste: A Constant Time Collaborative Filtering Algorithm. *Information Retrieval Journal*, 4(2), 2001.

[47] Object Management Group. CORBA Specification v3.0.3. http://www.omg.org/cgi-bin/apps/doc?formal/04-03-12.pdf. Accessed April 2004.

[48] E. Guttman and C. Perkins. RFC 2609 - Service Templates and Service: Schemes. http://www.ietf.org/rfc/rfc2609.txt. Accessed May 2004.

[49] E. Guttman, C. Perkins, J. Veizades, and M. Day. RFC 2608 - Service Location Protocol, Version 2. http://www.ietf.org/rfc/rfc2608.txt. Accessed May 2004.

[50] Erik Guttman. Service Location Protocol: Automatic Discovery of IP Network Services. *IEEE Internet Computing*, 35(4), 1999.

[51] Zoltan Gyöngyi and Hector Garcia-Molina. Web Spam Taxonomy. Technical report, Stanford University, 2004.

[52] Sayed Hashimi. Service-Oriented Architecture Explained. http://www.ondotnet.com/pub/a/dotnet/2003/08/18/soa_explained.html. Accessed April 2004.

[53] Sumi Helal. Standards for Service Discovery and Delivery. *IEEE Pervasive Computing*, 1(3), 2002.

[54] Sumi Helal, Nitin Desai, Varum Verma, and Choonhwa Lee. Konark - A Service Discovery and Delivery Protocol for Ad-hoc Networks. In *3rd IEEE Conference on Wireless Communication Networks (WCNC)*, 2003.

[55] Jonathan L. Herlocker, Joseph A. Konstan, Al Borchers, and John Riedl. An Algorithmic Framework for Performing Collaborative Filtering. In *22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, 1999.

[56] Todd D. Hodes, Randy H. Katz, Edouard Servan-Schreiber, and Lawrence Rowe. Composable Ad-hoc Mobile Services for Universal Interaction. *ACM Wireless Networks Journal, Special Issue on Mobile Computing: Selected Papers from MobiCom'97*, 5(5), 1997.

[57] T. Howes. RFC 2254 - The String Representation of LDAP Search Filters. http://www.faqs.org/rfcs/rfc2254.html. Accessed May 2004.

[58] An-Cheng Huang and Peter Steenkiste. Network-Sensitive Service Discovery. In *4th USENIX Symposium on Internet Technologies and Systems (USITS)*, 2003.

[59] Evan Hughes, David McCormack, Michel Barbeau, and Francis Bordeleau. Service Recommendation using SLP. In *IEEE International Conference on Telecommunications (ICT)*, 2001.

[60] ICSOC. 1st International Conference on Service Oriented Computing. http://www.unitn.it/convegni/icsoc03.htm. Accessed April 2004.

[61] ThoughtWorks Inc. Web Services: Pathway to a Service-Oriented Architecture. Whitepaper. http://www.thoughtworks.com/library/SOA.pdf. Accessed April 2004.

[62] Software Shelf International. Print Manager Plus. http://www.printmanagerplus.com. Accessed August 2004.

[63] John G. Kemeny. Mathematics without Numbers. *Daedalus*, 88, 1959.

[64] Maurice G. Kendall. *Rank Correlation Methods.* Charles Griffin and Company Limited, 2nd edition, 1955.

[65] Tim Kindberg and John Barton. A Web-Based Nomadic Computing System. *Computer Networks*, 35(4), 2001.

[66] Tim Kindberg and Armando Fox. System Software for Ubiquitous Computing. *IEEE Pervasive Computing*, 1(1), 2002.

[67] Mark Klein and Abraham Bernstein. Toward High-Precision Service Retrieval. *IEEE Internet Computing*, 8(1), 2004.

[68] Joseph A. Konstan, Bradley N. Miller, David Maltz, Jonathan L. Herlocker, Lee R. Gordon, and John Riedl. GroupLens: Applying Collaborative Filtering to Usenet News. *Communications of the ACM*, 40(3), 1997.

[69] Harumi Kuno and Akhil Sahai. My Agent Wants to Talk to Your Service: Personalising Web Services through Agents. In *1st International Workshop on Challenges in Open Agent Systems*, 2002.

[70] Peter Lacey. UDDI and Dynamic Web Service Discovery. *Dr. Dobb's Journal*, February 2004.

[71] Choonhwa Lee and Sumi Helal. Protocols for Service Discovery in Dynamic and Mobile Networks. *International Journal of Computer Research*, 11(1), 2002.

[72] Choonhwa Lee and Sumi Helal. Context Attributes: An Approach to Enable Context-awareness for Service Discovery. In *3rd IEEE/IPSJ Symposium on Applications and the Internet*, 2003.

[73] George Lee, Steven Bauer, Peyman Faratin, and John Wroclawski. An Agent for Interactively Learning User Preferences On-Line for Wireless Services Provisioning. In *3rd International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 2004.

[74] George Lee, Peyman Faratin, Steven Bauer, and John Wroclawski. A User-Guided Cognitive Agent for Network Service Selection in Pervasive Computing Environments. In *2nd IEEE Conference on Pervasive Computing and Communications (PerCom)*, 2004.

[75] Jonathan Levin and Barry Nalebuff. An Introduction to Vote-Counting Schemes. *The Journal of Economic Perspectives*, 9(1), 1995.

[76] Ryusuke Matsuoka, Yannis Labrou, Bijan Parsia, and Evren Sirin. Ontology-Enabled Pervasive Computing Applications. *IEEE Intelligent Systems*, 18(10), 2003.

[77] James McGovern, Sameer Tyagi, Michael Stevens, and Sunil Matthew. *Java Web Services Architecture*, chapter 2: Service-Oriented Architecture. Morgan Kaufmann, 2003.

[78] Robert E. McGrath. Discovery and Its Discontents: Discovery Protocols for Ubiquitous Computing. Technical Report UIUCDCS-R-99-2132, National Center for Supercomputing Applications, Department of Computer Science, University of Illinois, 2000.

[79] Sheila A. McIlraith and David L. Martin. Bringing Semantics to Web Services. *IEEE Intelligent Systems*, 18(1), 2003.

[80] Sheila A. McIlraith, Tran Cao Son, and Honglei Zeng. Semantic Web Services. *IEEE Intelligent Systems*, 16(2), 2001.

[81] Microsoft. Indigo. http://msdn.microsoft.com/Longhorn/understanding/pillars/Indigo/default.aspx. Accessed April 2004.

[82] Microsoft. Microsoft .NET. http://www.microsoft.com/net. Accessed April 2004.

[83] Microsoft. Universal Plug and Play Device Architecture (UPnP). http://www.upnp.org/download/UPnPDA10_20000613.htm. Accessed May 2004.

[84] Sun Microsystems. Java Remote Method Invocation (Java RMI). http://java.sun.com/products/jdk/rmi. Accessed April 2004.

[85] Mark Montague and Javed A. Aslam. Condorcet Fusion for Improved Retrieval. In *11th International Conference on Information and Knowledge Management (CIKM)*, 2002.

[86] Thomas P. Moran and Paul Dourish (editors). Introduction to this Special Issue on Context-Aware Computing. *Human-Computer Interaction*, 16(2-4), 2001.

[87] OpenSLP. The OpenSLP project. http://www.openslp.org. Accessed May 2004.

[88] Massimo Paolucci, Takahiro Kawamura, Terry R. Payne, and Katia Sycara. Semantic Matching of Web Services Capabilities. In *International Semantic Web Conference (ISWC)*, 2002.

[89] Mike P. Papazoglou. Service-Oriented Computing: Concepts, Characteristics and Directions. In *4th International Conference on Web Information Systems Engineering (WISE)*, 2003.

[90] Mike P. Papazoglou and Dimitrios Georgakopoulos. Service-Oriented Computing. *Communications of the ACM*, 46(10), October 2003.

[91] Edoardo Pignotti, Pete Edwards, and Gunnar A Grimes. Context-Aware Personalised Service Discovery (short paper). In *16th European Conference on Artificial Intelligence (ECAI)*, 2004.

[92] Thomi Pilioura, Aphrodite Tsalgatidou, and Alexandros Batsakis. Using WSDL/UDDI and DAML-S in Web Service Discovery. In *WWW 2003 Workshop on E-Services and the Semantic Web*, 2003.

[93] Stanislav Pokraev, Johan Koolwaaij, and Martin Wibbels. Extending UDDI with Context-Aware Features Based on Semantic Service Descriptions. In *International Conference on Web Services (ICWS)*, 2003.

[94] Shankar R. Ponnekanti and Armando Fox. Application-Service Interoperation without Standardized Service Interfaces. In *1st IEEE International Conference on Pervasive Computing and Communications (PerCom)*, 2003.

[95] Shankar R. Ponnekanti, Brian Lee, Armando Fox, Pat Hanrahan, and Terry Winograd. ICrafter: A Service Framework for Ubiquitous Computing Environments. In *3rd International Conference on Ubiquitous Computing (UbiComp)*, 2001.

[96] Salil Pradhan, Cyril Brignone, Jun-Hong Cui, Alan McReynolds, and Mark T. Smith. Websigns: Hyperlinking Physical Locations to the Web. *IEEE Computer*, 34(8), 2001.

[97] Easy Software Products. Common UNIX Printing System (CUPS). http://www.cups.org. Accessed August 2004.

[98] M. Elena Renda and Umberto Straccia. Web Metasearch: Rank vs. Score Based Rank Aggregation Methods. In *18th Annual ACM Symposium on Applied Computing (SAC)*, 2003.

[99] Michael D. Resnik. *Choices: An Introduction to Decision Theory.* University of Minnesota Press, 1987.

[100] John Riedl, Joseph Konstan, and Eric Vrooman. *Word of Mouse: The Marketing Power of Collaborative Filtering.* Warner Books, 2002.

[101] Frederic Schick. *Making Choices: A Recasting of Decision Theory.* Cambridge University Press, 1997.

[102] Bluetooth SIG. Bluetooth Core Specification v1.2. https://www.bluetooth.org/foundry/adopters/document/Bluetooth_Core_Specification_v1.2. Accessed May 2004.

[103] Michael Stevens. Service-Oriented Architecture Introduction, Part 1. http://www.developer.com/net/article.php/1010451. Accessed April 2004.

[104] Michael Stevens. Service-Oriented Architecture Introduction, Part 2, http://www.developer.com/net/article.php/10916_1014371_2. Accessed April 2004.

[105] Michael Stevens. The Benefits of a Service-Oriented Architecture. http://www.developer.com/net/article.php/1041191. Accessed April 2004.

[106] Oliver Storz, Adrian Friday, and Nigel Davies. Towards "Ubiquitous" Ubiquitous Computing: an Alliance with "the Grid". In *System Support for Ubiquitous Computing Workshop at the 5th Annual Conference on Ubiquitous Computing (UbiComp)*, 2003.

[107] Thomas Strang. Towards Autonomous Context-Aware Services for Smart Mobile Devices. In *4th International Conference on Mobile Data Management (MDM)*, 2003.

[108] James Surowiecki. *The Wisdom of Crowds.* Little, Brown, 2004.

[109] Katia Sycara, Massimo Paolucci, Anupriya Ankolekar, and Naveen Srinivasan. Automated Discovery, Interaction and Composition of Semantic Web Services. *Journal of Web Semantics*, 1(1), 2003.

[110] Doug Tidwell. Web services - The Web's next revolution. http://www-106.ibm.com/developerworks/edu/ws-dw-wsbasics-i.html. Accessed April 2004.

[111] Michel Truchon. An Extension of the Condorcet Criterion and Kemeny Orders. Cahier de recherche 9813, Département d'economique, Université Laval, 1998.

[112] Michel Truchon. Figure Skating and the Theory of Social Choice. Cahier de recherche 9814, Département d'economique, Université Laval, 1998.

[113] uddi.org. UDDI Technical White Paper. http://www.uddi.org/pubs/Iru_UDDI_Technical_White_Paper.pdf. Accessed May 2004.

[114] Efstratatios Valavanis, Christopher Ververidis, Michalis Vazirgianis, George C. Polyzos, and Kjetil Nørvåg. MobiShare: Sharing Context-Dependent Data and Services from Mobile Services. In *IEEE/WIC International Conference on Web Intelligence (WI)*, 2003.

[115] C. J. van Rijsbergen. *Information Retrieval. 2nd Edition.* Butterworths, 1979.

[116] Ellen M. Voorhees. Evaluation by Highly Relevant Documents. In *24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, 2001.

[117] Roy Want, Andy Hopper, Veronica Falcao, and Jonathon Gibbons. The Active Badge Location System. *ACM Transactions on Information Systems*, 10(1), 1992.

[118] Mark Weiser. The Computer for the 21st Century. *Scientific American*, 265(3), September 1991.

[119] Mei-Mei Wu and Diane H. Sonnenwald. Reflections on Information Retrieval Evaluation. In *1999 EBTI, ECAI, SEER and PNC Joint Meeting*, 1999.

[120] Dongyan Xu, Klara Nahrstedt, and Duangdao Wichadakul. QoS-Aware Discovery of Wide-Area Distributed Services. In *1st IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid)*, 2001.

[121] H. P. Young. Condorcet's Theory of Voting. *The American Political Science Review*, 82(4), 1988.

[122] W. Zhao, E. Guttman, and C. Bisdikian. RFC 3421 - Select and Sort Extensions for the Service Location Protocol (SLP). http://www.faqs.org/rfcs/rfc3421.html. Accessed May 2004.

[123] Feng Zhu, Matt Mutka, and Lionel Ni. Classification of Service Discovery in Pervasive Computing Environments. Technical Report MSU-CSE-02-24, Department of Computer Science and Engineering, Michigan State University, 2002.

[124] Feng Zhu, Matt Mutka, and Lionel Ni. Splendor: A Secure, Private and Location-aware Service Discovery Protocol Supporting Mobile Services. In *1st IEEE International Conference on Pervasive Computing and Communications (PerCom)*, 2003.