Année 2010

THÈSE

Présentée à
L'UFR DES SCIENCES ET TECHNIQUES
DE L'UNIVERSITÉ DE FRANCHE-COMTÉ

Pour obtenir le

GRADE DE DOCTEUR
DE L'UNIVERSITÉ DE FRANCHE-COMTÉ

Spécialité Informatique

Architecture de mécanismes adaptatifs pour offrir des services de confiance

A Framework for Adaptive Mechanisms for Trusted Services

Par

Oana DINI

Soutenue le 20 Décembre 2010 devant le jury :

Directeurs : H. GUYENNET, Professeur, Université de Franche-Comté
P. LORENZ, Professeur, Université de Haute Alsace

Rapporteurs : J. RODRIGUES, Professeur, Université de Beira Interior, Portugal
J. LLORET, Professeur, Université de Valence, Espagne

Examinateurs : Y.Q. SONG, Professeur, INPL, LORIA, France
A. ABOUAISSA, Maître de Conférences, Université de Haute Alsace, France

Acknowledgement

First, I would like to express my deep gratitude to my supervisors, Professor Hervé GUYENNET and Professor Pascal LORENZ, for their continuous support, guidance, and encouragement throughout the course of these three years. They helped me get the ideas across and clearly formulate the concepts presented here and provided me with the opportunity to discover and develop many interests.

My sincere thanks also go to the members of my dissertation committee: Professor Joel RODRIGUES, Professor Jaime LLORET, Professor Ye-Qiong SONG, and Professor Abdelhafid ABOUAISSA for their insightful comments and constructive feedback.

Finally, I would like to thank my family for their invaluable support. Special thanks go to my husband, Cosmin, whose encouragement and understanding helped me fulfill my goal, and to our two children, Theodora and Isabella, who tried their best to be patient and understand when plans had to be changed.

Abstract

The amount of information and services that are available over the Internet is so overwhelming that it is very difficult to select the ones that fit our requirements. Recommender Systems are being used for helping with the selection of services and products. The current approaches use product ratings to compute the reputation of a provider not taking into consideration the possibility of indirect service delivery. For this purpose, we propose a framework and appropriate mechanisms that can be used to better evaluate the reputation of services/providers. Another issue regarding the accuracy of a service reputation update comes from the dynamics of the feedback. Current approaches do not make a correlation with the users' feedback pattern or with the frequency of the users' feedback. We take these into consideration when computing a service reputation and propose an approach for modeling the dynamic user feedback. Service similarity is another important part of Recommender Systems. The customer's satisfaction depends on how close a recommendation is to his requirements, but also on how easy it was to specify the searching criteria. We present an approach for selecting services based on distance and similarity, introducing a similarity taxonomy for adjusting service invocation under certain constraints.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

QoS	Quality of Service
SLA	Service Level Agreement
ODP	Open Distributed Processing
ORB	Object Request Broker
LOTOS	Language of Temporal Ordering Specifications
XML	Extensible Markup Language
WS	Web Service
RS	Recommending Systems
IOS	International Standards Organization
ITU	International Telecommunications Union
RM-ODP	Reference Model International Telecommunications Union
IDL	Interface definition Language
OMG	Object Management Group
OMA	Object Management Architecture
CORBA	Common Object Request Broker Architecture
WSDL	Web Services Description Language
P2P	Peer-to-Peer
UDDI	Universal Description, Discovery and Integration
SOAP	Simple Object Access Protocol
HTTP	Hyper Text Transfer Protocol
STS	Security Token Service
CF	Collaborative Filtering
TF-IDF	Term Frequency/Inverse Document Frequency
MDP	Markov Decision Processes
RT	Role-based Trust-Management
PSPL	Portfolio and Service Protection Language
TPL	Trust Policy Language
TE	Trust Establishment
QoE	Quality of Experience
MOS	Mean Opinion Score
PESQ	Perceptual Evaluation of Speech Quality
I/O	Input/Output
ICRA	Internet Content Rating Association

CHAPTER 1

Introduction

1.1Motivation

The continuously evolving user needs and behavior, as well as the rapid development of offered services, with similar features, but at different service level agreements and/or quality of service raise the question of trusted services. The dynamics of user and services pose the trust issues, especially when services are invoked in different contexts, or users' perception is captured by specific feedback for appropriate service offerings. Current approaches make certain assumptions regarding feedback and information on the service provider identity, in order to easily compute the reputation of a service. They mostly base their reputation computation of a service/provider on the customer's feedback. Feedback can be fictitious, manipulated, all these contributing to the reputation update of a service/provider. A recommender should take these issues into consideration to correctly update the reputation, but so far the current proposals do not address them.

Searching for a service can still be difficult to do, even with the help of recommenders. There are many similar services available, but when it comes to having the user specify certain criteria when searching for a service (e.g., specify which service

feature is the most important and which one is optional) the options are limited. Also, many features are added to services/products without making the effort to evaluate the need for those additions. Consumers' feedback is important and it should be carefully evaluated before making any modification, additions to the current services/products.

1.2 Thesis Contribution

The thesis presents a framework for models and mechanisms for validating and enforcing trusted services based on the new concepts of progressive trust, context-based trust, and context-aware reputation. These mechanisms are used along with the user feedback composition to build a dynamically adaptive trust model for Web Services.

The basis starts with the reality that in current service selection, although there is a diversity of models of trust and recommenders there is a lack advanced mechanisms for (i) adaptive trust /context-trust/, (ii) context-aware reputation, and (iii) user feedback.

To handle the large spectrum of end-user behavior, we introduce specific notions and mechanisms that allow a more accurate service evaluation.

We introduce a framework and appropriate mechanisms to evaluate the services/providers in the light of their respective direct impact on user perception. Essentially, the proposal considers several innovative ways of considering user impact on an accurate evaluation of a service/provider reputation. The proposed schema can capture indirect service delivery and allow reputation correction based on the real transactions.

This is realized via an innovative mechanism for reputation computation including a context-based user model.

We propose an approach for modeling the dynamic user feedback when computing the reputation of a service. We introduce specific metrics describing the customer feedback, on the one side, and the variations in service transactions and customer subscriptions, on the other side. We show that the metrics can be used to specify different heuristics leading to a more accurate reputation (using dynamic updates based on feedback patterns). We propose a dual architecture to compute service reputation from both provider and customer views, and build a dynamic customer feedback model and appropriate heuristics to adapt the service delivery activities, accordingly. We show how this approach can be used to trigger appropriate actions to optimize service delivery under agreed QoS/SLA (Quality of Service/ Service Level Agreement) metrics.

Specific mechanisms are proposed to handle service similarities. It is still difficult to easily specify relevant ways to invoke a service. We propose a model and classes of similarity that can be used by a user to request and find a service that satisfies his requirements.

We show by statistics how the proposed mechanisms can be used in a real system and elaborate on its integration in current recommenders to add advanced features.

1.3 Structure of the thesis

Chapter 1 is a general introduction containing the motivation for this thesis and the thesis contribution.

Chapter 2 presents the state of the art related to Recommender Systems. It first discusses the evolution of the Recommender Systems and then it continues with more details regarding recommenders, trust, and reputation.

Chapter 3 presents a framework and appropriate mechanisms that can be used to evaluate the services/providers in the light of their respective direct impact on user perception.

Chapter 4 presents an approach for modeling the dynamic user feedback, when computing the reputation of a service.

Chapter 5 presents an approach for selecting services based on distance and similarity, introducing similarity taxonomy for adjusting service invocation under certain constraints.

Chapter 6 revisits some of the aspects presented in the previous chapters and presents a business-oriented platform model, a cost function to be optimized, and adjacent methods on how to settle and update service reputation in real cases.

Chapter 7 concludes the thesis discussing different correlations within the model we proposed and highlights certain aspects that need improvements through new solutions.

CHAPTER 2

State of the art

Summary

In Chapter 2, we present state of the art that is related to recommending systems. We start with the first proposal for trading services, which is ODP (Open Distributed Processing) Trader. The role of the ODP trader is to select the most suitable service based on a request. It has a trading function, which is used for searching services. The main components of the trader approach are importer (the party that issues the service request), exporter (the party that has the service), and service trader (the mediator). This proposal was too complex due to its type-based approach and had scalability issues.

The next proposal was the Object Request Broker (ORB), which was an improvement from the previous approach as it used the notion of object. The ORB would first find a server that could handle the method invocation, pass it to it and then forward the response to the client. To solve scalability issues, a protocol for ORB to ORB communication was proposed.

With the arrival of XML (Extensible Markup Language), a new approach for service discovery and service invocation was brought up, which was called Web Services (WS). The architecture of WS is similar to the previous approaches, but it has other advantages, such as platform and programming language independence. WS is defined as being an application that defines ways to make possible an interaction between different services that are running on different platforms and networks. There are certain risks that come with this approach and they are related to security and trust.

The latest approach used for recommending services is the Recommending System (RS). This is a system that collects ratings, analyzes them, and produces recommendations to user. There are several types of systems, each of the being defined by the methods used for producing the recommendations.

We conclude the section discussing the concepts of trust and reputation and their properties since these become more relevant in today's service interactions.

2.1 Trader ODP (Open Distributed Processing)

The paradigm of looking for a services based on a particular need was introduced decades ago; ISO (International Standards Organization) / ITU (International Telecommunications Union) focused on an architecture that supports distribution, internetworking, interoperability and portability. They developed a reference model (RM-ODP) covering all the phases of a system development captured in five views, which are: enterprise (purpose, scope and policies) information (semantics of information), computational (functional decomposition), engineering (infrastructure), and technology (choice of technology for implement).

While the ODP series of standards concerns the entire system design process, the core piece related to the services is the ODP Trader, which is a manger to trade services [1]. It supports a *trading function* used to search for services. All players in an open distributed system assume the roles of *service providers* and *service requesters*, having no *a priori* knowledge about each other. In particular they are not linked statically to each other, but rather dynamically when the need arises, by service invocation. The role of the trader is to select the best suitable service based on a given request. It appears that there is a need for semantic service type specifications, allowing the description of services with identical operational interfaces but different qualitative characteristics.

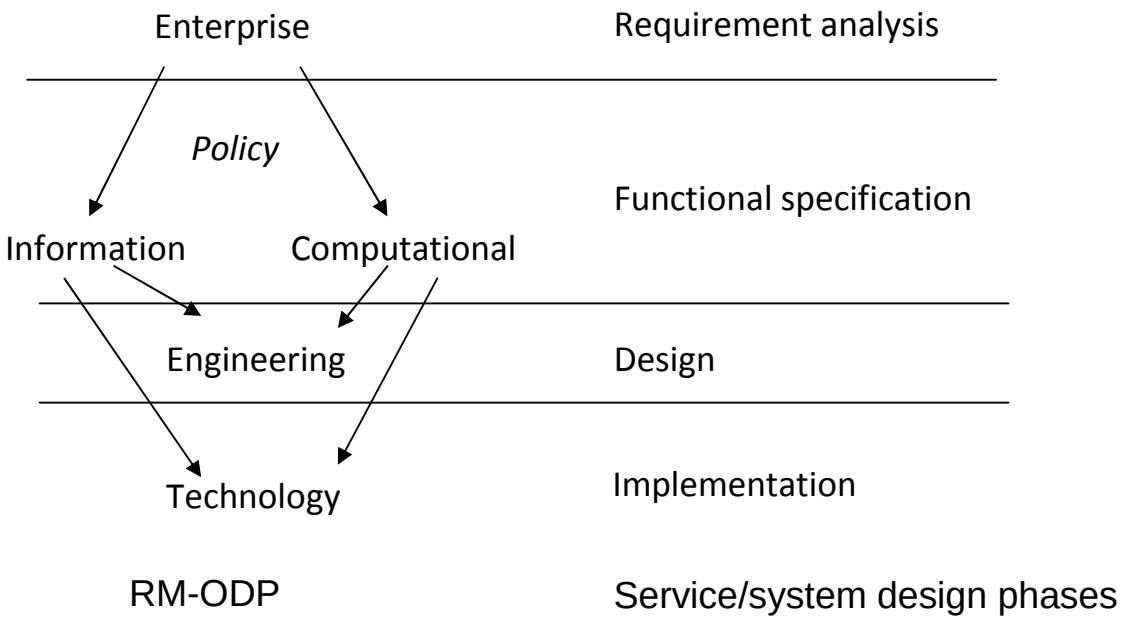


Figure 2.1. Mapping ODP Viewpoints and service development cycle

To evaluate the evolution of service trader approach to on-line shopping, the main concepts are presented in Figure 2.2.

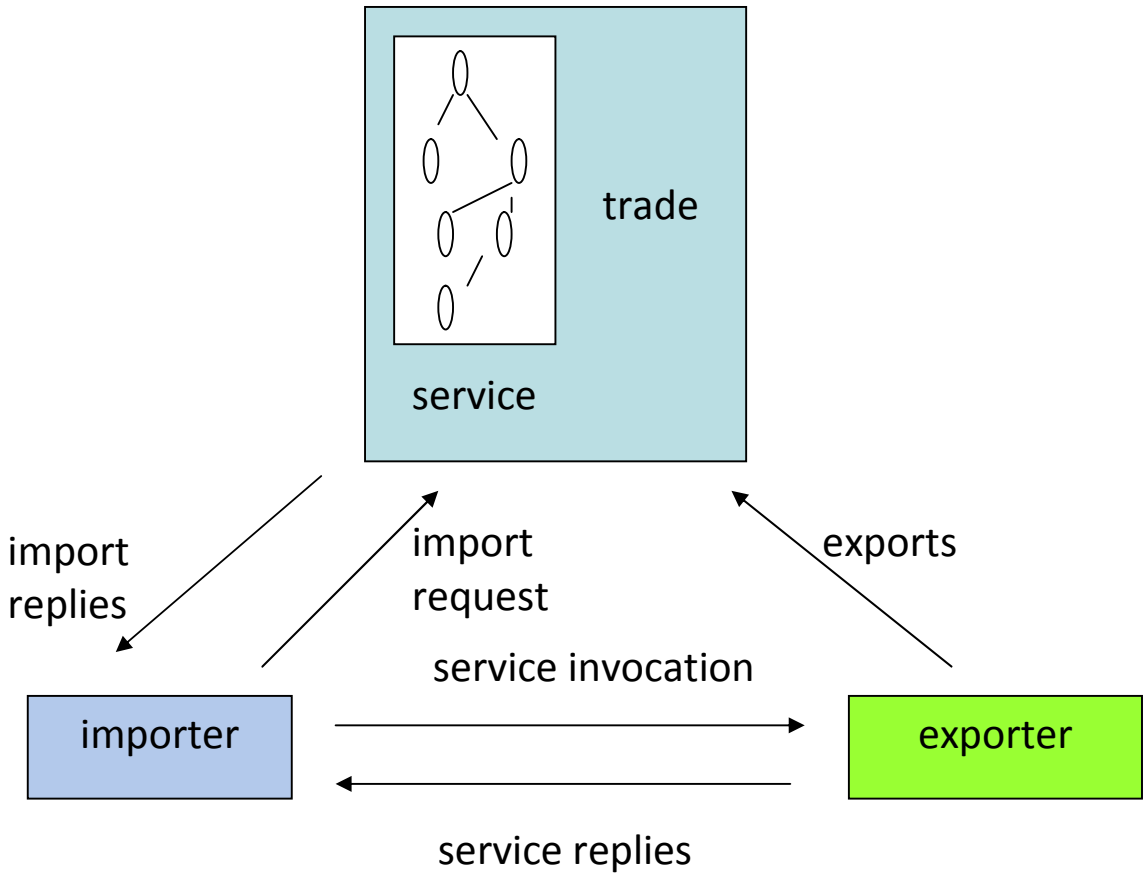


Figure 2.2. Exporters, importers, and service trader

Exporters register their services into a service space managed by a trader (acting as a mediator). Importers issue requests for a service; the trader searches in the service space for the most appropriate service and replies with the information for that service to be invoked. Two characteristics should be highlighted: (i) there is a request for formal specification of services (type) and (ii) after returning a service, the trader does not follow the importer/exporter interaction.

To satisfy the constraints on service type, a service hierarchy was devised (see Figure 2.3).

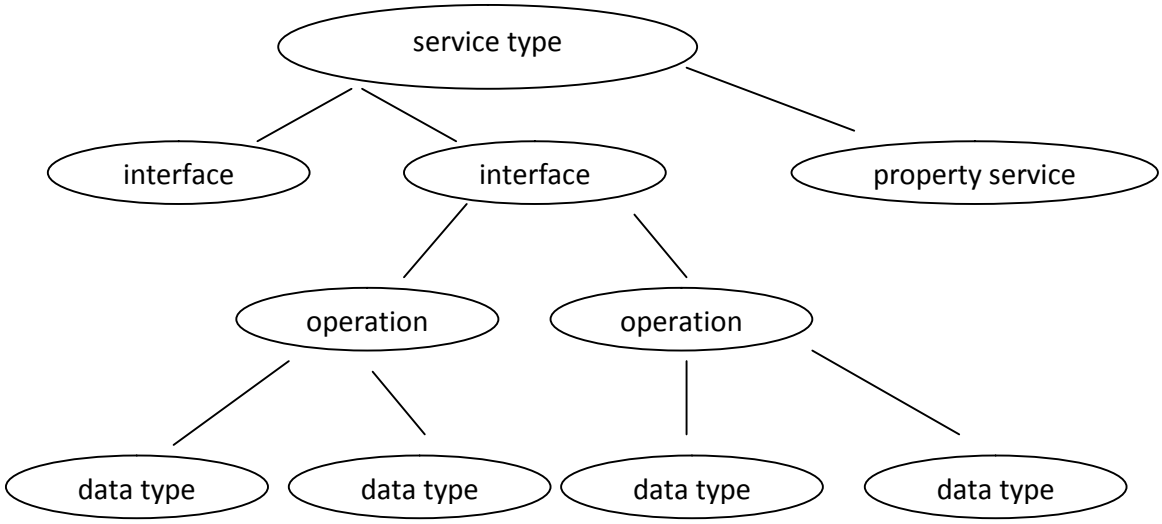


Figure 2.3. Service type hierarchies

Exporters should use a formal specification to correctly type their services and all information related to service operations and the input/output data types they operate with. The trader is situated at the enterprise viewpoint in the model. Each viewpoint has recommendations for the most appropriate specification languages. Standard specification for service interfaces was in IDL (Interface definition Language). The IDL compiler was quite complex. A lighter implementation was to use existing specification languages. Due to a type-based approach, many trader specifications were proposed using different formalisms, from LOTOS [100], Z[101], GMDO [102], Petri Nets [103], etc. While very accurate, this led to a high complexity and weak performance parameters. Scalability was also an issues, as some formalisms do not necessarily support it, e.g., Petri Nets. Also, mapping the specification to an implementation language (mainly C, Smalltalk; C++ later) was a barrier. IDL supports interface inheritance, but no details on

implementation. However, many systems were built following these concepts; the most notable was ANSAware [2].

2.2 OMG and CORBA

In late 90's, the concept of object and the era of object-oriented and object-based approaches drove the academia and industry. The OMG (Object Management Group) reconsidered the concepts and proposed an OMA (Object Management Architecture) having a bus-like trader called ORB (Object Request Broker) [3]. The main shift was using the notion of Objects and adopting a more popular specification (also implementation) language, Java.

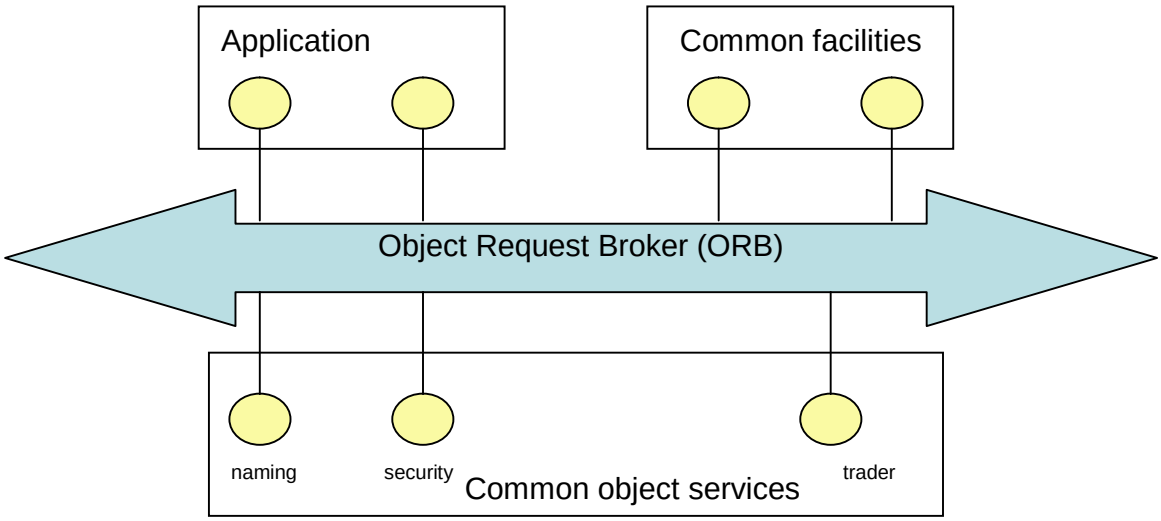


Figure 2.4. Object request broker

OPD's IDL cannot provide a full definition of ORB, as IDL is a declarative language for interfaces. The ORB is an abstract entity acting as the middleware for all

remote method invocations. The ORB finds a server that can handle a method invocation, passes the request to the server, receives the response and forwards it to the client. The functions handled by an ORB are actually implemented in both client and server. ORB handles implementation aspects, too, as shown in Figure 2.5.

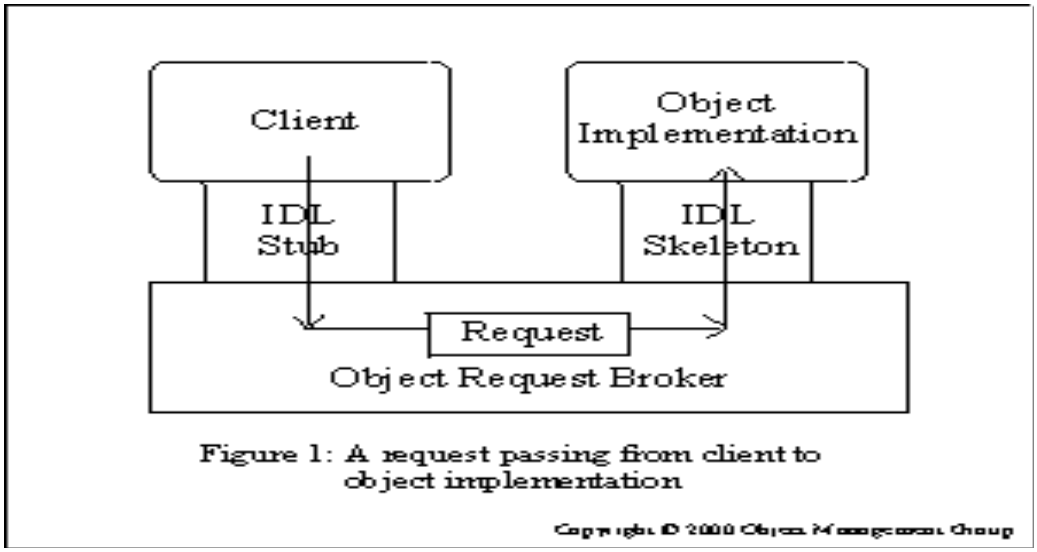


Figure 2.5. Handling object requests [4]

For scalability, a new inter-ORBs protocol was proposed, as shown in Figure 2.6.

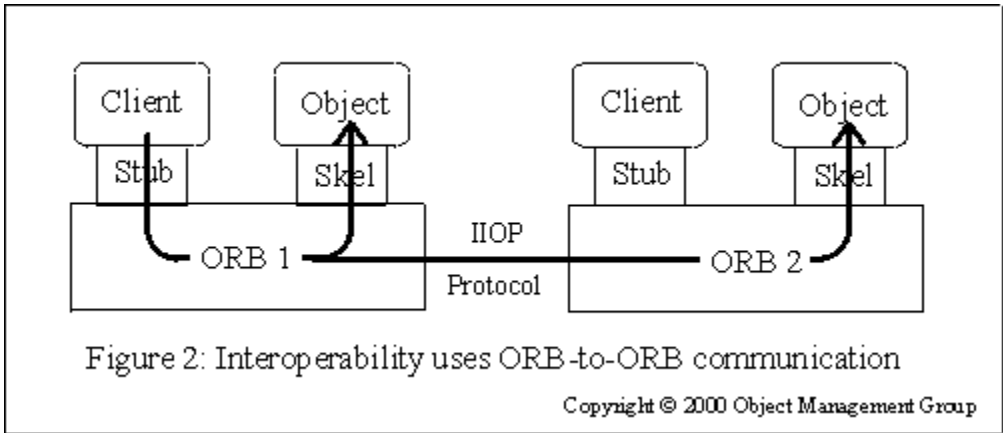


Figure 2.6. ORB-to-ORB communication [4]

OMG developed a large library with IDL-java mapping.

The advent of XML[104] produces a turn towards Web Services and allowed new approach for service discovery and service invocation. However, as we will see, there was not much deviation from the original principles. We stress out that no invoker feedback was considered so far in any approaches in order to optimize the trader/broker recommendations. Also, while the Trader considered only typed interface mapping, ORBs extended the decision on implementation aspects too. No quality of service for service delivery were considered, nor invoker feedback.

2.3 Web Services

The notion of Web Services (WS) describes a standard way to make interactions between applications that run on different platforms and different networks possible. A WS is an application that has three main properties: it can be discovered, described, and accessed [5]. The WS architecture is similar to previous architectures that have as the main components the service provider, service consumer, and service broker. From the figure below, one can see the similarity between WS architecture, Trader ODP architecture, and CORBA architecture. A service is published by a service provider trough a service broker. A service consumer finds a certain service by querying a service broker.

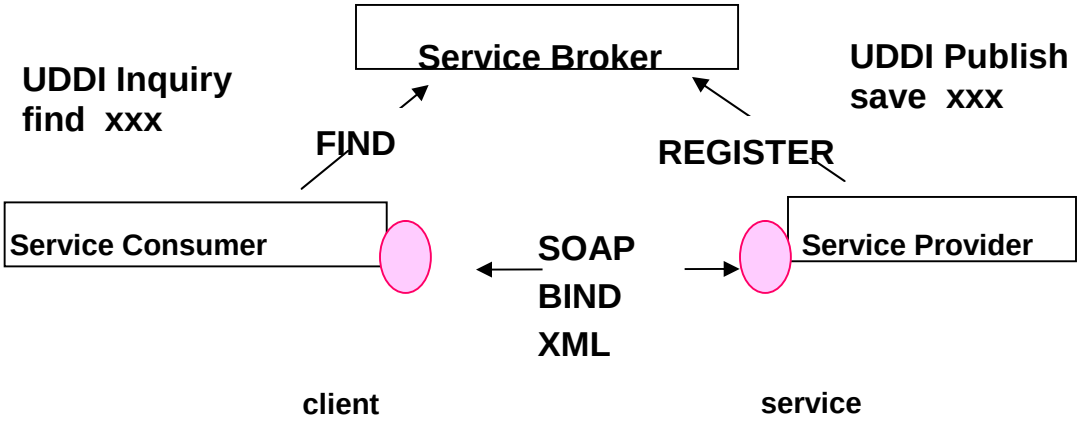


Figure 2.7. Web Services Architecture [6]

Here is a summary of the main activities necessary to operate a WS [5][7]:

- A WS has to be created and the interface and invocation methods have to be defined
- A WS has to be published, so users can locate it
- A WS has to be located, so the users can invoke it
- A WS has to be invoked, if a user wants to use it

The service description is done through Web Services Description Language (WSDL) [105]. WSDL uses XML grammar to describe what a service does, how a service should be invoked, and also the location of a given service. The information is used for making possible the communication between web services, following the same specification approach.

The discovery and publishing of a web service can be done using a registry, an index, or a peer-to-peer (P2P) system [8]. The registry approach is centralized and

controlled by its owner. The service provider has to put the information about the service into the registry. The owner of the registry decides who can put and update the information in the registry. For example, if one company wants to register a functional description of a different company, it won't be able to do it since the first company is not the provider of the service. The generally accepted reference for a registry is Universal Description, Discovery and Integration (UDDI) protocol [6][9]. The index approach is not centrally controlled and there is no central authority that decides who can publish what data. The data/services reside with the owner and the indexes just point to them without having to get permission from the service owner. The information on the index page may be out of date, but since the index points to the owner of the service, the interested user can verify the contents before using it. One index example is Google. The P2P approach used for discovering WS is also not centrally controlled. Whenever a query is received, a node can either respond with the requested information if they have it or they send the request to their peers, which can also propagate it. The benefit of this approach is that there is no single point of failure since it doesn't depend on a central registry, but it can be inefficient with a large overhead if the path to the resource is too long and also, it may not reach the entire network.

All of the above discovery approaches have their advantages and disadvantages, but the registry approach is still favored. Since there are multiple registries available, it would more efficient and useful to get information from more than one registry or index by querying what appears to be a single service. That is what Federated Discovery Searches do. They have information about other registries and indexes, and when a

request comes in, the query is also being sent to several indexes with probable relevant information. The results are pooled and presented to the user in a unified manner. For example, an air travel service may know about other registries that contain information on train travel services. When someone is searching for an itinerary, more than one option can be presented to him. In order to have an efficient communication between registries, it is advised to have the registries share a common ontology.

WS can be described, discovered, and invoked, but in order to communicate with a WS, a protocol is needed. SOAP protocol [106] (it used to stand for Simple Object Access Protocol) is used to encode requests and replies, using XML, which makes it programming language and platform independent. SOAP runs on top of the network and transport layer, HTTP being the most popular choice.

Communication between WS or between a user and WS does not come without any risks. Security and trust are two issues that need to be addressed. The security part has to do with the integrity of the communication between the components, while trust is related to the reliability of a specific component. To address the security part, certain conditions have to be satisfied. Since in WS both parties can be invoker and invokee, both have to authenticate themselves. Different WS may come from environments that have different security requirements, so the access control should depend on the content of the resource also. The integrity of the message is important, so for that matter, public key cryptography should be used. To satisfy message integrity, confidentiality, and security token passing, Web Services Security (WS-Security) was created. It is a specification that describes improvements to SOAP [10]. In order to establish a trust

relationship between the two parties, Web Services Trust Language (WS-Trust) was created. It defines mechanisms for requesting and emitting security tokens and also for managing trust relationships [11]. These mechanisms help the involved parties to decide whether to trust the other party's credentials or not. An example would be a client sending a request to a Security Token Service (STS) for a security token. The client sends this token to a service provider in order to obtain permission to use a service. If the service provider trusts the STS that emitted the token, then it allows the client to use the specific service. [6]

Compared to CORBA, whose components are tightly coupled, WS has loosely coupled entities and they do not depend on implementation details. While WS addresses trust issues, it only does this from the point of authentication. The reputation of the service and its impact on the user's perception is not considered in any of the above approaches. Also, none of them take into account the user's feedback after a service was used. When it comes to services discovery and selection, the users do not have the possibility to specify certain service features that they consider mandatory.

2.4 Recommenders

Recommender systems (RS) have been the subject of many studies and products over the last decade. The term was first brought up by Resnick and Varian [12], which, as mentioned in [13], it was mostly a replacement for "collaborative filtering" proposed in [14].

Recommender systems are defined as systems, which collect ratings from users and then analyze the data to produce recommendations to other users [15].

The framework below shows the main components and the general process of a RS.

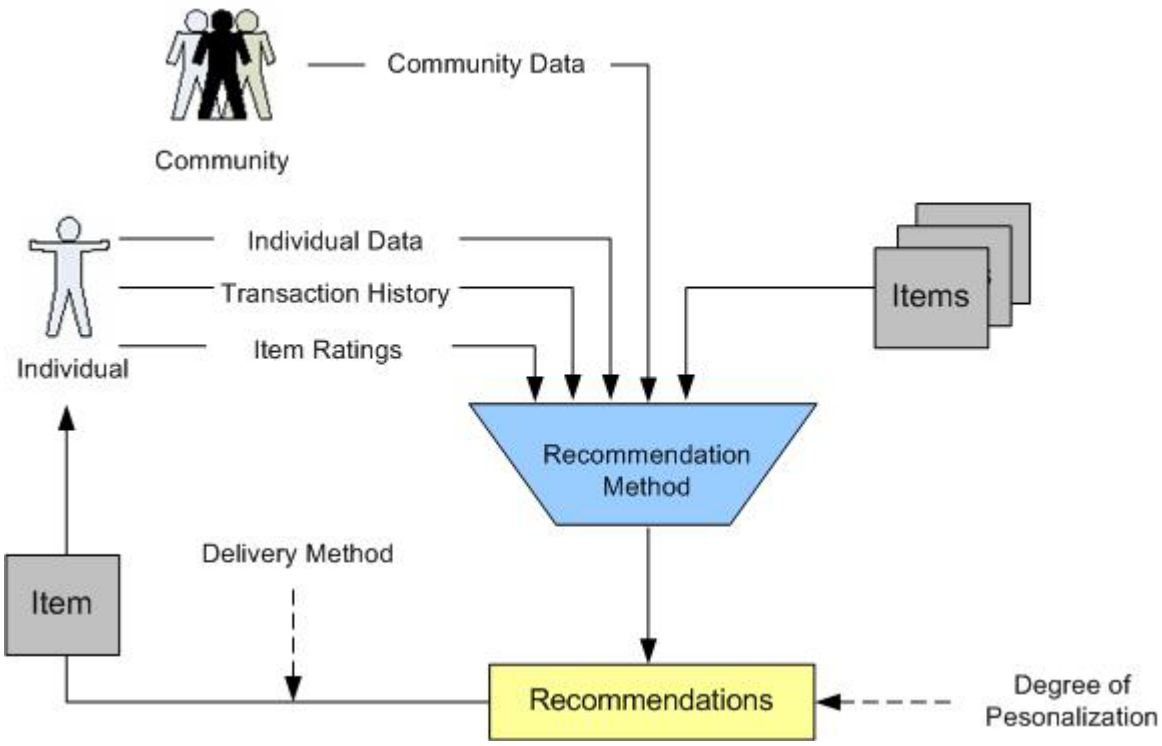


Figure 2.8. Recommender System generic architecture [16]

There can be several types of input data. The individual data is related to the user, such as gender, age, hobbies, occupation, etc. Based on this attributes, a RS can make certain recommendations. For example, if the user is a 15 years old girl, student, who likes skating, the RS may recommend certain books that are about skating and are

appropriate for the age of the user. The transaction history is an implicit type of preferences since it is inferred from the user's behavior such as browsing, shopping, likes and dislikes. To continue the previous example, if the student is browsing and looking at a book about figure skating, the RS may suggest similar items (Amazon usually does that whenever a customer clicks on an item). The item ratings notion is exactly what the name implies: the user's ratings, which are obtained either explicitly by asking the user to review a product or service, or implicitly by watching the user's behavior. We will see a bit later how recommenders try to predict similarity between users. Community data refers to the general opinion of a community about certain products or services. For example, statistics can be collected to obtain the best seller list within the community or a compilation of the products that have the highest ratings can output a list of the most popular products.

There are several techniques used to generate recommendations, but the main categories are *Content-based Filtering*, *Collaborative Filtering (CF),* and *Hybrid approaches* [17].

2.4.1 Recommenders Categories

The *Content-based Filtering* recommends to users items that are similar to the ones searched by the users in the past [17][18]. This type of recommendation technique is mostly used to recommend text-based items such as documents and newspapers. In order to produce the recommendations, the system needs a profile of the user, which is

represented by a set of terms. The profile can be obtained from the user through a questionnaire or it can be learned from their past transactions. The user's profile has his preferences, needs, and, over time, ratings for the previously visited items. The ratings can also be obtained directly from the user by having the user rate the relevance of the item or indirectly through observing the user's actions. The latter approach is more convenient for the user, but it's harder to implement and it also takes time to determine the relevance of a recommended item. The relevance of an item is calculated using the weight of the important words. The Fab system [19] represents documents in terms of the 100 most important words. The importance of the words is calculated using term frequency/inverse document frequency (TF-IDF) [20]. The words that occur more frequently in a document are considered to have more weight for the topic of the document. In Syskill & Webert [21] the approach is slightly different. The authors make use of the 128 most informative words, which in their case are the words that are closer related to a certain class of documents (e.g., "protein" in a biomedical document).

This type of filtering has its shortcomings. Since it is content-based, it needs to have the representation of data in a matter that can be machine-parsable (e.g., article). It is harder to apply this technique in the case of movies, music, images, which are not machine-parsable. Descriptions of this type of media can be inserted manually, but it is not very practical (time consuming), it is easy to make mistakes, and hard to find the right keywords that would classify the item properly. [18] For example, the word bracket can have different meanings such as parenthesis in a formula, support for a shelf, dental bracket, and architectural bracket. Also, in order for the system to make a proper

recommendation, the user has to go through many items and rate them so that a more accurate profile of the user can be obtained and used in recommendations. Going back to the bracket example, a user needs to read through a number of articles before the system can figure out the type of bracket the user is interested in.

The *Collaborative Filtering* (CF) [14] tries to predict the relevance of an item based on the ratings done by other users. It accumulates ratings of products and whenever a request comes, the system identifies similar users and recommends the products rated by them. In this type of filtering, the user profile is defined by a vector of items and their ratings, which is updated over time. There are three main algorithms used within the CF: *memory-based algorithm, model-based algorithm, and hybrid algorithm* [13].

The *memory-based algorithm* uses statistics on the entire database of ratings to find customers with similar tastes. The similarity measure can be computed in different ways, but the two most popular approaches are the correlation-based and cosine-based techniques [17][18]. The correlation-based approach uses the Pearson correlation coefficient to measure the similarity between two users with respect to their rating of a service or product. Both users need to rate some of the items in order to compute the coefficient. The cosine-based approach considers the users as vectors and measures the similarity by computing the cosine of the angle between them. This algorithm is popular because it is simple, easy to implement, and quite effective [14][22]. Two of the most well known systems that used memory-based CF algorithm are Amazon and Barnes and Nobles.

Even though the memory-based CF algorithm is popular, it does face a number of limitations. One limitation is data sparseness. The algorithm needs to have enough common items rated by the users in order to calculate the similarity between users and make effective recommendations based on that. The approach is also not scalable enough when it comes to large datasets due to resource requirements. Since this algorithm does not build a model based on the user's profile, there is not much information or insight learned from the profile [14].

The *model-based algorithm* builds a model of user ratings and uses it to make predictions. This approach uses machine learning and data mining algorithms to build the model. The most used techniques are the Bayesian network model, clustering model, Markov decision processes (MDPs), and latent semantic [23]. In the Bayesian model, each node of the graph is a variable and each directed arc is a probabilistic association between the nodes. The structure of the network and the probabilities are learned from the date. This type of model is usually used for classifications [17]. In the cluster model, similar users are grouped in clusters and the recommendations are made within the cluster, which makes the approach more scalable. In the MDP model, Shany *et al.* [24] view the recommendation process as a sequential optimization problem instead of a prediction problem. The state of a recommendation is the response of the user to that recommendation (e.g. taking the recommended item, not taking it at all, taking a different item). This approach is usually used in applications where the user can influence its surroundings. The latent semantic model is based on statistical modeling techniques that include latent class variables in order to discover user communities and prototypical

interest profiles. In [25], the authors decompose the users' preferences by overlapping user communities. The benefit of this approach is that it helps in discovering patterns and regularities that show the users' interests and also their disinterests.

The *hybrid algorithms* usually combine the content-based and the collaborative algorithms to overcome some of the limitations of the other two approaches. This approach has been adopted by some RS [26][27][28]. Other kinds of hybrid approaches are *demographic-based RS,* which uses information such as the user's location, occupation, gender, *utility-based RS* and *knowledge-based RS,* which use knowledge regarding the user's satisfaction with a certain object [23][29].

There are different ways to combine the two main CF systems. One way is to add content-based features to CF models. The Fab CF recommender [30] maintains user profiles using content-based approach and uses CF approach to find similar users. By doing this, it can recommend items across profiles and it also help with the sparsity related problems that are present in a pure CF system. A similar approach is presented in [31], where artificial agents called "*filterbots*" act as participants within a CF community. The users whose ratings are similar to those of the filterbots receive better recommendations. The *content-boosted* algorithm [32] performs better than a content-based recommending or a CF recommending. Since the user matrix is usually sparse, the content-boosted approach uses a denser matrix called *pseudo ratings matrix,* which contains the user's ratings and the ratings predicted by the content-based predictor for the items that were not rated by the user.

Another way for obtaining a hybrid recommender is to combine different RSs. The results from each recommender can be combined into one recommendation [17] using different techniques such as linear combination of ratings [33], weighted majority voting [34], or weighted average voting [35]. Choosing a single recommender instead of a combination of recommenders is also an option. In [36], the recommender with a higher level of confidence is selected, whereas in [37], the chosen recommender is the one that was more consistent with the past ratings of a user. Other approaches offer a method to combine the collaborative and content-based recommendations into a single probabilistic framework [26]. The model extends Hofmann's approach [25] and includes three-way co-occurrence data among users, items, and item content.

Hybrid algorithms can provide better recommendations when compared to a content-based or a pure CF method, but at the same time, they are more complex and not easy to implement.

2.5 Trust

With the increasing use of social networks, Web-Services, and on-demand services, the concept of trust becomes more relevant. In order for a user to make use of a service, it needs to establish a trust relationship with the service provider. Since trust has different meanings from one user to another, due to their personal experiences and their expectations, establishing a trust relationship is not an easy task. Even the definition of trust varies among sources. The Webster dictionary defines trust as "*An assured reliance on the character, ability, strength, or truth of someone or something.*", or *"A charge or*

duty imposed in faith or confidence or as a condition of a relationship." or *"To place confidence (in an entity)."* This definition is more of an intuitive definition that can be used within a social context, but too generic to adopt it within Web Services context. [38] proposes a definition that fits better within the Web Services domain; trust is" the quantified belief by a trustor with respect to the competence, honesty, security, and dependability of a trustee within a specified context". In [39], the authors capture the notion of trust from two different perspectives defining it as *reliability trust* and *decision trust. Reliability trust* can be understood as the dependence and reliability of the trustee as perceived by the trustor. *Decision trust* is the degree to which, given the circumstances, one is willing to depend on an entity, with the understanding that less than satisfactory outcomes are a possibility.

2.5.1 Trust Properties

The notion of trust is usually presented within the context of a relationship between parties. There are several properties of trust that are discussed in literature [38][40][41]. One property is the subjectivity of trust. Different individuals may link the same experience to different levels of trust. For example, what is perceived by Alice as very trustworthy may be viewed by Bob as only trustworthy. One study [42] presents a method for eliminating the subjectivity from trust recommendations. Their approach is to use a percentile instead of an absolute value for reporting the trust level. The reason for this is that an absolute value does not have the same meaning for different individuals

because of their difference in the disposition to trust. If Alice transmits a value of 6 to Bob as the trust value for Ellie, Bob does not know if this value means a high trust level or an average one. If the value were sent as a percentile, such as 60%, Bob would know that Alice considers Ellie to have an average trustworthiness. The local value that Alice has for Ellie is not relevant to Bob. Bob can change the percentile that he received from Alice to his own local values that have meaning for himself.

Transitivity or non-transitivity is another property that has been a subject of debate. Some authors suggest that trust transitivity should not be considered because it can lead to unintentional transitivity [40][43]. If Alice trusts Bob and Bob trusts Ellie, Alice should not automatically trust Ellie without Bob's consent. If transitivity is accepted, then the effects of unintentional transitivity should be taken into account when analyzing the level of trust. In [44][45], the authors address these problems by limiting the delegation depth and by decreasing the trust along the recommendation paths. In [46], the authors describe semantic criteria and conditions for specifying a valid trust transitive network. In order to build such a network, only personal experience (e.g., direct trust) should be referred, every edge of the network has to have the same "trust purpose" (e.g., find a good doctor), and the last edge is the "functional trust" (e.g., being a good doctor).

Trust is not monotonic, meaning that the level of trust changes over time due to one's own experience as well as extraneous experiences. Since trust is not uniformly applicable in all instances and situations, the level of trust should be considered within a specific context. For example, a student may trust the mathematics professor with a math question, but not with a Biology question. Even within a context, the trust level can

change over time based on experience and expectations. Trust is a dynamic mental state [47]. The dynamic nature of trust has been and still is an issue that affects the mechanisms for trust computation and the trust models. Jonker *et al.* [47] mention that the factors that influence someone's trust can be divided in two categories: factors related to the evaluation of an event and factors related to the weight of an experience. The first factor depends on the context where the event took place and the second factor is related to the experiences that a user had over time with someone or something. An example for the first factor would be a customer driving a new car and after a short time, the car breaks down. The disappointment in the product will be much bigger than if the customer were driving a used car. The expectations were higher for the new car. In the case of the second factor, the weight of an experience is related to the time that it passed since the event happened. Someone who had an unpleasant experience with one car a year ago will have more trust in it then in a car that broke down a week ago.

In [48], there are several factors related to the dynamic nature of trust: actual behavior, expected behavior, willingness, capability, context, time, association type. These factors are almost self explanatory, with the exception of the last one, association type. The association type takes into account profiling information based on interaction between entities (truster, trustee, and groups that contain either of these two).

2.5.2 Trust Management

One of the first ones to define trust management were Blaze *et al.* [49], and this was done for the purpose of helping with the verification of security policies. Since it mostly focused on managing public keys authorizations and not on trust management [50] added to the definition the idea of collecting security related information, analyzing it, and presenting it for the purpose of decision-making. Grandison [38] went even farther with the definition by extending the idea of collecting security information to collecting information related to competence, honesty, and dependability for the purpose of making decision regarding trust relationships.

The main two approaches for trust management are *policy-based trust management* and *reputation-based trust management* [41].

2.5.2.1 Policy-based Trust Management

Policy-based trust management focuses on managing mechanisms and policy languages that are used for specifying rules for trust establishment. In order to establish a trust relationship between two parties that have no common transaction history, certain credentials and policies have to be satisfied. Credentials are usually related to authentication, access control, integrity, and privacy [6]. In a classic client-server system, only the requester is authenticated by the provider, but in this case the authenticity of both parties has to be ensured, a notion that is called *trust negotiation*. An example would be when the requester sends sensitive information, such as credit information, to the service provider. In this case, both parties should sign their messages in order to

ensure the integrity of it. The credentials of both parties are disclosed gradually so that no sensitive information is revealed before a trust relationship is established [51].

Access control is usually on a per user basis, but the access has to be associated with the content also, and with the conditions that are associated with the provided content. The access control does not depend only on the user's identity, but on the content of the resource also. The resource-owner allows access to the requested resource only if the requester's credentials can be verified either directly or through a series of trusted agents. [41][52]

Privacy of the exchanged messages and the integrity of their content are important for guaranteeing the preservation of expected quality of service. The contents of a message should only be read by the requester for whom it is intended. Since most of the time the message has to pass through more than one intermediate point, the information pertaining to trust and the actual message should be encrypted, but the routing information needs to remain without encryption [52]. To ensure that an unauthorized third party has not modified the received messages, public key cryptography should be used. The sender signs the message with its private key and sends it to the other party. The receiver verifies the signature by using the sender's public key. This approach provides non-repudiation [53].

The trust negotiation process is automated and usually done by security agents on behalf of the users, which makes it transparent to users [51]. There are several policy languages used for trust negotiation such as Role-based Trust-Management (RT) [54], [55], Portfolio and Service Protection Language (PSPL) [56], and Trust Policy Language

(TPL) [57]. These languages have to meet certain requirements, which the authors in [51] separate them in two categories: language expressiveness and semantics and runtime compliance checker. The involved parties have to be able to conclude if the credentials satisfy their policy. To do that, the policy language has to be clear, concise, and to have a well-defined semantic. To increase the level of trust, policies should require a combination of credentials so that in the case of an attack, the attacker will need to compromise more than one key in order to get access. To express complex conditions, a policy language needs to have the expressive power of a query language and transitivity closure. The authentication requirements need to be either part of the language or external function calls. By doing this, the party that submitted the credentials will have to prove at runtime that it knows the private key associated with the public key that was specified in the credentials. Certain policies can also be sensitive and for that matter, they need to be protected. If for example, a policy states who can access an account, then outsiders can conclude certain information regarding that account and/or the owner of the account.

The RT framework is used to represent policies and credentials in distributed authorization [54][58]. The framework consists of several languages that have the same basic structure. These languages are RT_0, RT_1, RT^T, and RT^D. RT combines the qualities of role-based access control and trust management systems making it appropriate for attribute-based access control. It uses the notion of role, which is a set of principles. The main purpose of the roles is to give to the members access to specific resources. For example, the local library gives discounts to students from university A. A student, who

is not from university A, tries to get a discount at the local library, but the request is denied because he is not eligible for it.

PSPL is a language that is used for specifying access control policies for services and disclosure policies for client and server. It also contains a policy filtering mechanism, which helps in protecting the privacy during the policy disclosure [51][56]. The language is part of a larger framework that gives the client the possibility to offer counter-requests to the server and to enforce restrictions on information release. Within this framework, the interacting parties have a *portfolio* of credentials and declarations associated with them. In order to access certain data, the requester has to submit its credentials and declarations that are part of its portfolio. There is a difference between *data declarations* and *credentials*. Data declarations are statements expressed by one party (e.g., address, name), while credentials are statements that are also signed by a trusted authority (e.g., digital certificates). Other than these differences, the PSPL differentiates between rules that are used to access services and rules used to release portfolio information.

The TPL is a language that is part of the Trust Establishment (TE) system. It is an XML-based language that is used for mapping *entities* to *roles* [57]. The mapping is done using credentials that are emitted by a third party. Roles, which can also be called groups, are represented by the tag <GROUP>. Under roles, there are rules that apply to the members of the group. Each rule tag can contain only one rule. In order to become a member of a group, only one rule needs to be satisfied. Constraints within the rule can be specified by using the tags <INCLUSION> and <FUNCTION>. The <INCLUSION> tag specifies basic constraints on credentials (e.g., certificate issuer). There can more than

one <INCLUSION> tags under one rule. Under the <FUNCTION> tag, more constrains related to the certificates can be specified. One drawback of this TE system is that it does not protect sensitive credentials. It is assumed that the credentials can be released whenever they are requested. To protect this kind of information, the authors in [51] propose adding a function called *requester authenticates-to*, which would have as input the main public key. The other party will have to respond with the corresponding private key.

2.5.2.2 Reputation-based Trust Management

Reputation-based Trust Management deals with mechanism that can be used by the requesting party to evaluate resources based on their own experience with the provider and also based on the recommendations that they received from others. These mechanisms are necessary for building trust relationships within the online community [59][60]. For example, Alice wants to buy a used card from Bob on eBay. Since Alice cannot have the car inspected, she needs to base her decision on the description given by Bob. Alice doesn't know Bob so she needs to find out how trustable Bob is when it comes to business transactions. For this, Alice resorts to assessing his trustability based on feedback and comments already received from previous transactions with other users. Depending on the feedback, which makes up Bob's reputation, Alice can make a decision regarding the transaction. Given this, one can conclude that it is important for Bob to keep a good reputation so that he doesn't lose current buyers or prospective buyers, thereby increasing his customer pool. Several systems have been proposed, some of them

being centralized reputation systems, such as eBay and Amazon, some being decentralized, such as DMRep [61], EigenRep [62], and PeerTrust [63].

Reputation Properties

Reputation is sometimes confused with trust, when it is actually a factor that can affect the trust relationship. As mentioned in [64], reputation is a "perception of the trustworthiness of an agent based on experiences and recommendations". They define reputation of an agent as a "perception regarding its behavior norms, which is held by other agents, based on experiences and observations of its past actions."

Like trust, reputation is subjective, context dependent, time-sensitive, and dynamic. Reputation subjectivity is similar to trust subjectivity. Identical behavior can leave different impressions on different agents or recommenders. That is because each recommender has its own gagging mechanisms. What is very trustworthy for one agent or recommender may be only trustworthy for another. In [65], the reputation subjectivity is calculated by using a weighted mean of the agents' impressions. By impression it is meant the subjective evaluation of an agent done for a specific experience. In calculating the reputation subjectivity, the impressions that are recent have more weight. Also, to increase the level of reliability, it is necessary to have a larger number of impressions when computing.

The context dependency of reputation is another property that is important when evaluating one's reputation. For example, a brand of a car can have a great reputation for being reliable, but not a great reputation for gas consumption. Different context-

dependent reputation models have been presented such as those in [65][66]. Others base their model on the opinion collected within a context group [67]. In a context group, which is a union of agents that rate each other's behavior, the assessments are made for a specific context. PeerTrust model [63] takes into consideration, among other factors, transaction context and community context for evaluating the reputation. Sometimes, there may not be enough information within one context, but there is enough information in a similar context. In order to conclude a reputation based on the similar context, the distance between contexts has to be measured. [64] proposes a formula for computing the reputation on context for which there was not enough information. Reputation can also be transferred from one context to another [68]. An example would be inferring that one is a good speaker, good at reading people, good strategist, all of this because he has a reputation of being a good lawyer.

The dynamic property of reputation refers to the ability of an agent to modify its reputation; the reputation gets better if the service provided is consistently good or gets better with time; the reputation decreases if the provided service is inconsistent in quality or if the agent is dishonest. The values of an agent's reputation can change over time due to its behavior. Depending on the evaluation method used, the reputation can increase at different speeds. Many approaches attribute more weight to recent behaviors then to past ones [64][67].

Reputation Systems Overview

Reputation systems are mainly classified into *centralized systems* and *decentralized systems*.

Centralized reputation systems collect feedback ratings and store them in a central database. The ratings are processed by the system and a reputation value is produced and made publicly available. The centralized reputation systems have three main characteristics [69]: a) a central authority collects all the feedback and manages it; b) the reputation of a user is calculated by the system; c) to find out the reputation of one party, the inquirer only needs to communicate with the central authority of the system.

In eBay [70], which is one of the most popular centralized reputation based systems, the buyer and the seller might provide feedback after the transaction is complete. The feedback can be positive (+1), negative (-1), or neutral (0). One can also leave comments, especially if explanations are necessary, but they are not parsable so they cannot be part of statistics. The reputation of a user is calculated by aggregating the feedbacks over the past six months. An eBay membership comes with two distinct contexts: buyer and seller. eBay does not readily distinguish between feedbacks coming from the two different contexts. An approach to reputation building for new users is to participate in several small value transactions as a buyer, where the reputation has less of an impact. By receiving positive feedback on these transactions, reputation is gained from a single context, but in the end it reflects on both contexts in the eyes of prospective buyers.

Feedback manipulation is another issue that can affect reputation systems. eBay for example, doesn't have a problem with feedback manipulation. The feedbacks can only be left by users who are registered with them and who made a purchase on eBay. However, if a group of users agree with a seller to leave positive feedback for fictitious

auctions (e.g., the seller can post multiple 1 cent auctions on which the users can bid), the seller's ratings can be positively affected. These users are usually called shills. This approach would require quite an effort (the larger the number of shills, the bigger the impact) and also, it can cost the seller a substantial amount since eBay charges for every transaction.

Decentralized reputation systems do not use a central database to store the feedback ratings. Therefore there isn't a global view of the users' reputation so the users have to rely on each other to get information about the trustworthiness of a certain individual. Each user has to calculate the reputation of another user, store it, and manage it. Based on this, one can infer four characteristics for decentralized reputation systems [69]: a) there is no central authority to collect and manage the feedback ratings; b) each user assesses the trustworthiness of another user based on its personal experiences with that user; c) no global view of the user's reputation is available; d) to compute someone's reputation, the interested user has to exchange many messages with his trusted peers.

Several decentralized reputation systems have been proposed, each of them having certain benefits and drawbacks. The proposal in [40] has the benefit of having only for values for trust (very trustworthy, trustworthy, untrustworthy, and very untrustworthy). The drawback is that it requires that each user/agent keeps a quiet large data structure that depicts a global view of the network. Updating this data structure can be time consuming and not very scalable. The proposal from [61] does not require the user to keep a large data structure. It is based on binary trust, meaning that a user/agent can only have two values, trustworthy and not trustworthy. Users can forward their

dissatisfactions to other users and store them in a P-Grid data structure. When a user wants to assess someone's trustworthiness, it searches the leaves of the P-Grid for complaints on that user. This proposal is more scalable due to the storage and retrieval of data approach. It does not flood all the peers that are part of the system with query about other peers. One of the drawbacks is that a user/peer needs to keep information that is owned by other users/peers and this information cannot be modified. Another drawback is that if a user/peer leaves the system, some important information may be lost. Another proposal is XenoTrust [71], whose purpose is to model, administer, adapt, and distribute trust between members in the XenoServer Open Platform. XenoTrust has two types of trust: authoritative trust and reputation-based trust. The reputation-based trust is built through peer interactions based on individual experiences. The system accommodates new users by exchanging reputation information between peers. The reputation information is stored in the XenoTrust instead of the users. This approach has the drawbacks of the centralized systems since the reputation information is managed by XenoTrust, but it also has the advantage of not losing information if a user/peer leaves the system.

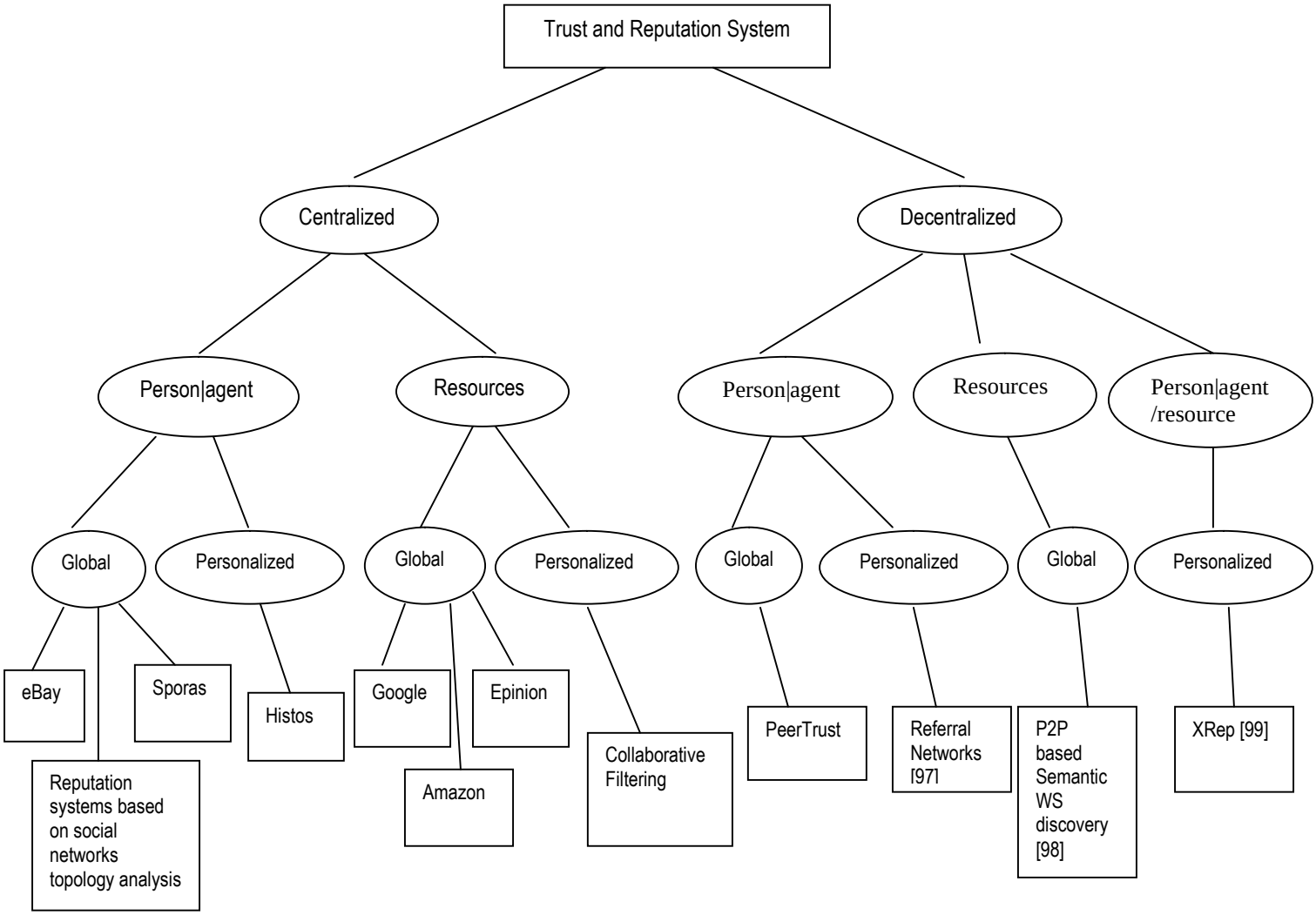


Figure 2.9. Trust and reputation system classification (adapted from [69])

In [69], the authors classify the trust reputation systems in more details, breaking it down in a three layer hierarchical manner [see Figure 2.9]. The first classification level, centralized vs. decentralized, is done based on the architecture used for managing the reviews and ratings. The second level splits each node based on the type of entity targeted for review: person/agent or resource/service. The classification of the next level is based

on the type of reputation. The reputation can be global, which means that it was build based on the reviews of the general population, or it can be personalized, which means that it was build based on the reviews of a specific group of people. For each type of reputation, the authors give examples of reputation based systems that meet their criteria.

Weather a reputation-based system is centralized or decentralized, there are advantages and disadvantages in both types. The centralized systems are simpler, which means easier installation, but because of the central authority that manages all the date, a reliable server is necessary. Other disadvantages are scalability issues, vulnerability, and flexibility. The decentralized systems are more complex. The reputation information is not stored in a central node, but it is spread among users/peers. This raises other issues that are related to reputation retrieval, evaluation, and propagation.

Recommender Systems became an important part of today's electronic commerce. Regardless of mechanism used for making recommendations, there is almost always a business model behind it. In [72], the authors present an analysis of the RS revenue structure.

Table 2.1 Analysis of Recommender System Revenue Structures [72]

SYSTEM OPERATOR		SYSTEM	REVENUE STRUCTURE	PAYER
		Amazon customer comments	Improve product sales	Product buyers
		CircuitCity customer review	Improve product sales	Product buyers
		Elance customer review	Improve transactions done Improve subscription base	Product and service providers Service providers
		eBay feedback profile	Improve transactions done Improve advertised items	Product and service providers Product providers
Third Party	Operators with offline publications	Zagat.com	Subscription fees Advertising fees	Advertisers (not product providers)
		ConsumerReports.org consumer survey	Subscription fees	Subscribers
	Operators that use the online channel only	CitySearch.com	Referral fees	Product sellers
		Shopping.com Epinions.com	Ad, pay-per-use, referral fees Sponsorship fees Content, software licensing	Product sellers Sponsors Aligned partners
		ConsumerReview.com	Ad referral, pay-per-use fees Sponsorship fees Content license fees	Product sellers Sponsors Aligned partners
		Shopzilla.com + BizRate.com	Referral fees Software, content licensing	Product sellers Aligned partners

Their approach is based on the framework proposed by Resnick and Varian [12] to which they apply the ideas regarding revenue structures proposed by Novak and Hoffman [73]. They classify the business model based on the type of operator system. The operators can be a sales agent/distributor or third party operators, which can use online channels or offline publications.

CHAPTER 3

An Enhanced Framework for Web Recommenders

Summary

Chapter 3 presents a framework and appropriate mechanisms to evaluate the services/providers in the light of their respective direct impact on user perception.

The web-based transactions, web services, and service oriented platforms require mechanism to announce, select, and use different services. There is a dilemma of ‘use and trust’ or ‘trust and use’ for different services based on the notion of reputation. Indirect servicing makes it difficult to really assess a given service or provider. A provider may be subcontracting certain services and not give the credit to the actual service provider. The question is how to make the recommender aware of these underlying transactions between the providers.

We propose an enhanced model that takes into consideration the user’s profile and behavior, plus a list of potential providers for a given service. This permits for a more accurate ranking scheme where the providers can be rated per service. It is shown that the proposed mechanism allows capturing situations usually not possible to be captured in the current approach. The technique has a major impact on e-commerce systems, on systems based on service-oriented architecture, and on all auction-based transactions.

3.1 Introduction

With the overwhelming amount of information, products, and services available over the Internet, it has become harder for the users to select the ones that fit best their needs or requirements. First of all, it is too difficult and time consuming to sort through hundreds of items and select the needed one. Also, there is the problem of trusting the provider for that item and not only that, but trusting that the provider is offering a product that meets the user's requirements. In order to assist the user in selecting the product or service that it needs, recommender systems (RS) have been proposed.

RS are important in electronic commerce, especially for marketing [74] and they have been widely used in order to attract and retain customers. The relation between the loyalty of users and RS was studied in [75] using data from Amazon.com. Their findings showed that the presence of consumer reviews helps with retaining customers and also attracting new ones. In time, the business gains reputation, which usually translates to increase in business.

There are a few challenges in optimally using the recommenders due to the variety of user's profile and its volatility and the reputation of different service providers. For dealing with these aspects, recommenders usually use product rating, confidence in service providers, and regularly update this information for an accurate suggestion for a given request.

In all the existing approaches, some unprovable assumptions are considered for the purpose of easily computing reputation. Aspects like partial feedback, ignorance of

customer confidence, and most importantly lack of information on the service provider identity are major challenges for an accurate reputation per product, per service provider, per context, per user profile.

In this chapter, we propose an approach taking into consideration the above challenges and deriving mechanisms for a more accurate reputation considering direct and indirect product delivery.

The chapter is organized as follows: Section 3.1 presents basic concepts and major achievements on recommender implementations. The proposal of an enhanced framework for an accurate reputation is presented in Section 3.2. A use case is presented in Section 3.3, while conclusion and future work are discussed in Section 3.4.

3.2 Related work

As the proposed approach touches the recommendation and reputation on recommenders, service providers, and products, we first recall some basic concepts.

3.2.1 Concepts

The core information of a recommender is a list of offers (products) and ratings of those products based on feedback received after a series of recommendations. The *rating* is subject to incomplete, fictitious feedback, volume of transactions for a given product or provider, and confidence in feedback. Based on the ratings, the recommender computes its own *ranking* per product.

s[r], P[r] represents a service or a provider with the rank r, where r is an integer.

Associated with the ranking is the notion of *reputation*, that in fact, determines the ranking. The reputation formula, while product oriented, it might not be accurate, as its computation cannot avoid some realities, such as some service providers have private relationships with recommenders (e.g., publicity, sponsorship) or indirect servicing (recommended product might not be produced by the front end provider, but simply delivered by it).

Reputation is an index associated with the service or a product based on user feedback that is taken into consideration when the ranking is calculated. The reputation index usually belongs to a **set, {outstanding, very good, good, acceptable, bad}.** A recommender might increase the rank of a service when its reputation index, for example, passes from very good to outstanding [76].

Similarity is another concept used in generating recommendations. In order for a recommender to suggest products to a user, it needs to find a commonality among users (this applies in the collaborative approach) or among the products that were rated in the past by the user (this applies in content-based approach). There are different techniques used to compute the similarity measure, but the most used are correlation-based and cosine-based techniques [17][18]. Similarity is an index associated with two services or products. For example, $\mathbf{s_1}$ **[~/80%]** $\mathbf{s_2}$ means s_1 is similar with s_2 with an acceptance of 80% based on the service's features or in the same range of ranking.

3.2.2 Current approaches for recommenders

Recommenders are usually classified based on the approach for making the recommendations. There are three main categories of recommender types: Content-based Filtering, Collaborative Filtering, and Hybrid Filtering. We recall here the types of recommenders giving a short overview of each type.

The *Content-based Filtering* recommends to users items that are similar to the ones searched by the users in the past [17][18]. This type of recommendation technique is mostly used to recommend text-based items such as documents and newspapers. In order to produce the recommendations, the system needs a profile of the user, which is represented by a set of terms. The profile can be obtained from the user through a questionnaire or it can be learned from their past transactions. This type of filtering has its shortcomings. Since it is content-based, it needs to have the representation of data in a matter that can be machine-parsable (e.g., article). It is harder to apply this technique in the case of movies, music, images, which are not machine-parsable.

The *Collaborative Filtering* (CF) [78] tries to predict the relevance of an item based on the ratings done by other users. It accumulates ratings of products and whenever a request comes, the system identifies similar users and recommends the products rated by them. In this type of filtering, the user profile is defined by a vector of items and their ratings, which is updated over time. As opposed to the CBF, this type of filtering can be applied to any kinds of items, not only to machine-parsable items. However, there are

limitations with this approach, mostly caused by the lack of data points in initial stages: new user and new item.

The *hybrid algorithms* usually combine the content-based and the collaborative algorithms to overcome some of the limitations of the other two approaches. This approach has been adopted by some RS [27][28]. There are different ways to combine the two algorithms and [17] present the different approaches in detail.

Like we mentioned above, the reputation of a business is gained in time, mainly based on reviews from users. This brings up another point and that is obtaining accurate reviews from users. Many users are not willing to leave feedback after a transaction is completed. One reason for not leaving feedback is the lack of incentives. If there isn't some kind of payoff for the feedback, the user won't put the effort into posting one. An incentive mechanism is addressed in [77], where incentives are given to users who provide honest feedback through a side payment mechanism. Examples of incentives mechanisms are Amazon's "Top Reviewers" practice and Epinions.com referral fees practice [72]. Another reason for not leaving feedback is to purposely withhold information about a product that gives its user an advantage [78].

Another concern related to the validity of the reviews is the manipulation of the reviews by parties with direct vested interest. Businesses can review their own products in order to boost the sales. Also, the competition can leave or fabricate negative feedbacks to undermine the competitor's reputation. There are ways to filter out biased feedbacks and to prevent manipulation [79], but preventing coordinated collusion attacks is still an issue. Like we mentioned in the previous chapter, eBay does not have a

problem with feedback manipulation. The feedbacks can only be left by users who are registered with them and who made a purchase on eBay.

Reputation is very useful in RS and eBay is one example of a reputation system that proves that their approach works well. However, having a centralized reputation system such as eBay can bring other issues, such as vulnerability and inflexibility of the system [78].

In [78], the authors propose a distributed trust and reputation management framework. The users choose a trust broker and after each transaction with a service, the user sends its rating to its trust broker. This way, the trust broker builds a reputation about a service based on the user's feedback. The brokers exchange reputation information among themselves in order to collect more information about the available services. This framework relies on the user's feedback only, ignoring the business model of the provider.

In reality, a provider may subcontract the service from somewhere else and in the end take all the credit. The question now is how to make the Recommender aware of the underlying transactions among the providers so all providers receive fair rating. If Provider 1 contracts a service from Provider 2, Provider 2 should receive credit for its service also.

3.3 An enhanced recommender model

In this section, we present a Recommender Model that can handle the sub-contract mechanism, yet keeping an accurate information on a given provider reputation (leading to an accurate ranking).

3.3.1 Setting the case

A simple scenario is presented in Figure 3.1, where the user is interested in service s_1 from P_1. The user asks the Recommender for the best provider for service s1 within specific parameters. The Recommender replies with either a provider that has the best reputation for service s_1 or with a list of providers $\{P_i\}$ for s_1. Let us assume P_1 is registered of being capable to deliver s_1 (others might be registered for s_1 as well). The Recommender cannot know if P_1 has the service or if it contracts it from a different provider. If P_1 is contracting s_1 from P_2, the transaction between P_1 and P_2 is not transparent to both the Recommender and the user. At the end of the transaction, the user sends the rating of P_1 to the Recommender and P_1 receives all the credit for the transaction. This leads to an inaccurate reputation and altered ranking.

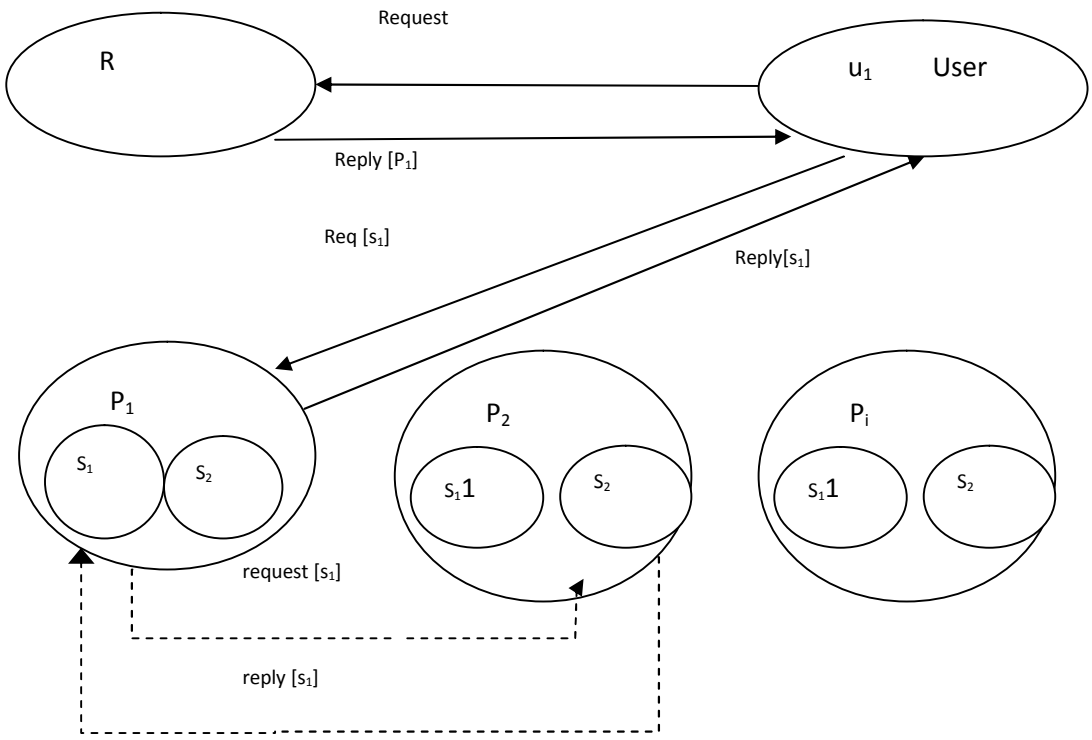


Figure 3.1. Indirect reputation

If the reputation of the provider is based only on the user's feedback, there is no way to assess the ultimate role of each provider. In order to have a more accurate picture of the providers' involvement, we propose that feedback from the providers be taken into account when establishing reputation. This includes both the front end provider (in our case P_1), as well as any subcontracted providers (in our case P_2). All feedback goes directly to the Recommender.

The ideal scenario would be when all the users and providers report 100% of the transactions. In reality, users don't always leave feedback and providers don't always report rendered services. In such a case, the Recommender is left to deal with an

incomplete set of data. Moreover, some of the reported data may be fabricated by both users and providers.

3.3.2 Recommender representation model

Apart from the mechanism of collecting the feedback and interfacing with the users, the core information present in a recommender is stored in a service database. This allows a request to be replied to with a service or a list of services, eventually with a degree of similarity associated with each service. Usually, the recommender keeps information on relative ranking among these entities.

We propose an enhanced model, which takes into account the user's profile and behavior, and a list of potential providers for a given service. This allows a more refined ranking scheme where providers can be rated per service.

While ranking is based on user feedback, there is no appropriate mechanism to consider the user's expectation (e) and credibility (c). By user expectation we mean the probability of having the user leave feedback after a service was delivered. The credibility refers to the user's ability to give a trusted rating. Usually both, expectation and credibility are expressed as percentage.

In Figure 3.2, we present the enhanced recommender model. The recommender stores information about the available services, the providers and their services, plus the user profile, which includes its expectancy and credibility. Both services and providers

are associated with a rating. The providers' rating is done within the context of a service. This way, the rating can be done per product and per provider for a specific product.

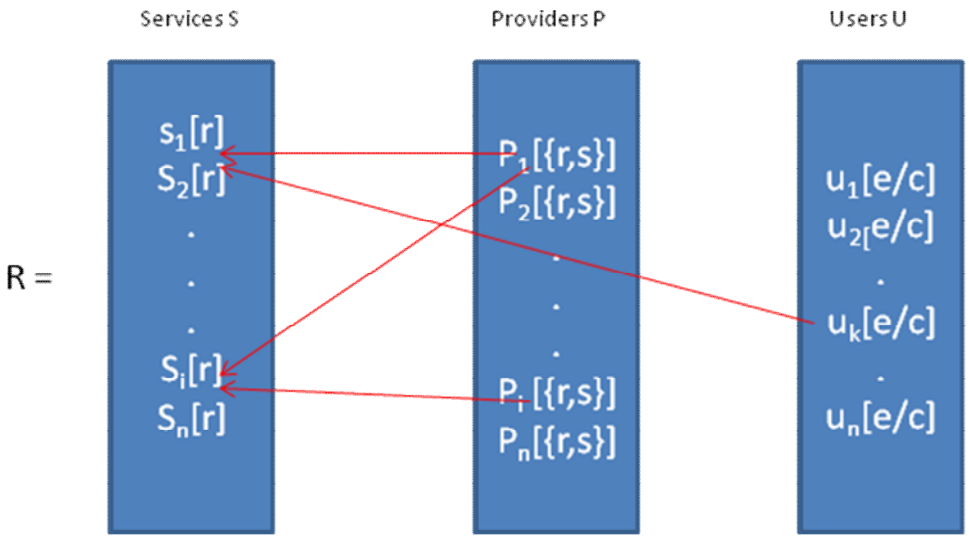


Figure 3.2. Enhanced Recommender Model

By keeping the relationships between the providers, their services, and also the users who requested the available services, the recommender can provide better suggestions and answer to more complex queries.

We classify queries in two categories, i.e., U-R and P-R. Some salient queries U-R might be:

Query 1:

input: [s_1]

output: [s_1/P_1, s_1/P_2]

The user asks for service s_1 and the recommender replies with a list of providers that offer s_1.

Query 2:

input: [s_1] & [s_1 (~/Ɛ)]

output: [s_1/P_1, s_1/P_2] & [s_i/P_i]

The user asks for service s_1 and/or a service similar to s_1. The recommender replies with a list of providers that offer s_1 and/or a list of providers who offer services similar with s_1.

"~/Ɛ" represents the similarity of services with Ɛ as proximity

---

Query 3:

input: [s] [P_1, P_2]

output: [s_1/P_1, s_2/P_1] [s_i/P_2, s_j/P_2]

The user asks for a list of services offered by certain providers. The recommender replies with a list of services offered by those providers.

Query 4:

input: [s | r > x]

output: [s_1/r_1, s_2/r_2]

The user asks the recommender for a list of services, which has a ranking "r" higher than a certain value. The recommender replies with the list of services.

Some relevant queries P-R might be the following:

Query 5:

input: [u_i]

output: [u_i [e/c]]

The provider asks the recommender about user u_i. This may be relevant to the provider in order to assess the user's credibility. The recommender replies with the u_i expectation "e" and credibility "c".

Query 6:

input: [all U_i, e > α, c >β]

output: [u_i [e/c]]

The provider asks the recommender for a list of users whose expectation and credibility are higher than a certain value. This may be relevant to the provider in order to assess the user's credibility. The recommender replies with the list of user(s).

Based on the formula presented in the following section, complex information can be gathered and more accurate answers to different queries can be provided.

3.3.3 Computation mechanism

The enhanced model allows a more comprehensive schema for computing the reputation.

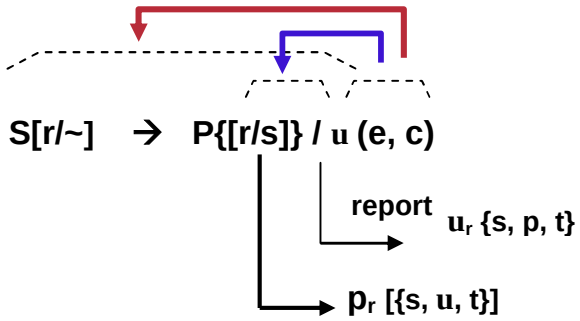


Figure 3.3. A computation schema for recommenders

In our framework, a recommender has mechanisms for representing services (S) with their reputation (r) and similarities (~), provider (P), with their reputation (r) linked to the reputation of their service providers (s), associated with user's (u) expectation (e) and credibility (c). A particular relation is valid at a moment (t). For example, a user x is expected to provide feedback with e = 80% and the confidence on its feedback is 70%. The feedback is on a provider (p) providing a service (s) at the time (t). The schema allows having a reputation view of a user at a given time, on a given provider delivering a given service. The schema also allows having a reputation of a provider, as perceived by a user at a given time, if delivered by a given provider.

We are now going to concentrate on different scenarios dictated by the amount of data reported by users and service providers.

For example, a user sends a request to the Recommender for the best cell phone provider that would meet certain parameters. The Recommenders replies with provider P_1. The user makes a request for a number of cell phones from P_1. After the transaction is completed, all the involved parties have the option to send feedback to the Recommender. The Recommender collects the data and based on the feedback, it updates the reputation of the involved parties. The nature of the collected data can be divided in three main cases:

1. Matching reports

The number of feedback reports from the user matches the number of reports from the service provider within a particular time window relevant to the service type. To continue with the example from above, the user sends the feedback to the Recommender, including the number of cell phones that it purchased. P_1 reports to the Recommender that the user purchased a number of cell phones from it. The numbers reported by both the user and P_1 match.

A subclass of this scenario would be when P_1 sub-contracts from a different provider, P_2. If P_1 receives a request for cell phones, it can send the products from its own stock, send part from its own stock and part from P_2, or get the entire order from P_2. In this case, the Recommender would receive reports from both providers, P_1 and P_2. The exact number reported would not match since P_1 will report that it sent the entire order to the user, and P_2 would report that it sent a certain number of phones to P_1, but the data

can be correlated. The correlation is done by using the transaction completion time, the user identifier, and the provider identifier.

2. Over-reporting providers

The number of feedback reports from user and provider does not match. This can be caused by either providers exaggerating the amount of transactions completed, or by users who underreport. In this case, some of the data can be correlated by the Recommender.

3. Underreporting provider

The number of feedback reports from user and provider does not match. This can be caused by either providers that do not report every transaction, or by users who exaggerate the amount of transactions completed. In this case, the Recommender can correlate some of the date.

3.4 Case study for reputation correction

Let us consider the following situation:

u -> [t] [p1] [s1], where u is the user, p1 and p2 are providers, t is the time of the request, and s1 is the service:

p1-> [t] [u] [s1], with p1 [r1/s1]

p2 → [t] [p1][s1], with p2 [r2/s1]

and the following transaction reports:

|u|: reports α transactions

|p1| reports β transactions (with β< α)

|p2| reports γ transactions

then

$k = (\beta - \gamma) / \alpha$ (3.1)

In this case, for a given user u, and for the considered service s1, the real reputation is $r_1' = k \times r_1$, as there is an indirect service delivery from p2 via p1 to the user u. The schema allows having a more accurate view on who is delivering a service.

Note: the number of transactions can be either reported or obtained by audit. In this use case, we consider that the providers are subscribed to an automated transaction report when delivering a service.

3.5 Discussion

In this section, we are comparing existing recommender systems with our proposal, on the basis of three main features: expectation, credibility, and user profile, as defined in Section 3.2.

3.5.1 Feature-based comparison

We consider a few well known recommender systems and only selected those three main features as a basis of comparison. The existing recommenders do not incorporate in the user profile the expectation and credibility of a given user.

Table 3.1. Feature based comparison of several recommender systems as well as the proposed one

	eBay	Amazon.com	Barnes & Nobles	proposal
expectation	Not in profile	Not in profile	Not in profile	Included in profile
credibility	Not in profile	Not in profile	Not in profile	Included in profile
user profile	yes	yes	yes	yes

While the considered systems (eBay, Amazon.com, Barnes & Nobles) make use of the notion of profile when recommending a product, the main target is to identify potential similar services and products to either satisfy a request or recommend a particular service unknown to the user (using the similarity concept).

By including these features, the recommender can have a more complete view on user's satisfaction based on more accurate information maintained by the system on the user's behavior (the degree of responsiveness of the user ability to give trusted rating).

3.5.2 Performance and accuracy

The performance and accuracy of a recommender system can be enhanced by including in the user's profile the user's expectancy and credibility. By having the expectancy of a user to leave a review and also its credibility, a recommender can better tune its suggestions to a user's requests with increased certainty. Ongoing experiments will identify the thresholds from where these features increase the accuracy of recommendations. Particular consideration will be given to the dynamics of user's

feedback in terms of relationships between the frequency (volume) of the used services or products and the accuracy of the timely feedback.

3.6 Conclusion

This chapter presented a framework and appropriate mechanisms to evaluate the services/providers in the light of their respective direct impact on user perception. Essentially, the proposal considers several innovative ways of considering user impact on an accurate evaluation of a service/provider reputation. The proposed schema can capture indirect service delivery and allow reputation correction based on the real transactions.

In the next chapter, the consistency feedback and reliability will be correlated with the frequency of users' report and transaction peaks, as well as with the user's report patterns. This will allow detecting of potential 'off-market' agreements between providers and set an appropriate service level agreement policy.

CHAPTER 4

Dynamic Feedback for Service Reputation Updates

Summary

Chapter 4 presents an approach for modeling the dynamic user feedback, when computing the reputation of a service. The interaction between every service provider and its users is regulated by the service level agreement and customer satisfaction feedback. The former is the basis for the technical audit, while the latter subjectively validates the user perception. To accurately evaluate the feedback after service/product consumption, we refine the user profile by considering the dynamics of the feedback. The challenge is to identify the appropriate metrics for computing the updated reputation. We present specific metrics that describe the customer feedback, on the one side, and the variations in service transactions and customer subscriptions, on the other side. We show that the metrics can be used to specify several heuristics that lead to a more accurate reputation. The approach we propose deals with peaks in feedbacks. We consider quick negative and quick positive feedback as well as late vs. early feedback with respect to the time of the transaction. We also formalize the seniority of responder as it might be new or old consumer. Considering a large scale measurement view, we formalize the feedback patterns on given services (for a given user or as average for a service/product) and show their relevance in accurately evaluating the service providers.

4.1 Introduction

Classically, two notions concerning the quality of a delivered service are correlated for an accurate service evaluation, i.e., QoS (Quality of Service), and QoE (Quality of Experience). Specific to each service, there are particular service parameters that are agreed upon between a provider and a subscriber, commonly settled by the SLA (Service Level Agreement). On the provider side, the SLA parameters are used for technical audit and litigations (leading to penalties or bonuses towards a given user or class of users). Specific on-line and off-line measuring mechanisms for SLA metrics and specialized audit techniques have been proposed. On the consumer side, the subscribers' satisfaction is gathered and mapped to the audit results to validate a given service, to detect flaws in delivering a service, and to ultimately build a view on service reputation. In general, a record is handled per service or per products, with respect to a given subscriber or a class of subscribers. Customer feedback can be 'by request', or 'at will', and embraces various forms of on-line questionnaires. As a result, a customer might decide not to answer, or to answer exhibiting a particular behavior. Ultimately, a service provider might fake some feedbacks to increase its reputation. There are various factors that influence the computation of an accurate reputation, e.g., the volume of ordered services, the diversity of the subscriber classes, customer trust and loyalty, and the dynamics of the feedbacks. Practically, the main problem we try to find a solution for is to dynamically and accurately compute the reputation of a service/product, based on the

system transactions. We propose a simple formula for reputation updates (4.1). The challenge is to identify the correct metrics in computing the updated reputation.

$$r_{real} = (1+ \lambda) \times r_{current} \quad (4.1)$$

where:

r_{real} represents the updated reputation, considering

$r_{current}$ represents the known and accepted reputation, and

λ represents the correction based on customer perception and feedback behavior, λ belongs to $\{(-\alpha, \alpha) \mid \alpha > 0\}$, usually having values in the vicinity of '0'.

The main achievement of our proposal is that the recommenders systems have a powerful mechanism to accurately indicate the real reputation, when selecting the best service provider from a service directory. Whereas most of the studies on QoE [80][81] mainly consider the technical metrics, we introduce and evaluate the customer behavior.

The chapter focuses on dynamic aspects of customer feedbacks and formalizes mechanisms for a more accurate reputation evaluation of a given service delivered by a given provider in establishing policies for λ. Section 4.1 presents related studies. A taxonomy of the dynamic feedback and a dual architecture are presented in Section 4.2. Sections 4.3 and 4.4 introduce computation mechanisms for an accurate reputation of a service via dynamic reputation update policies and heuristics. We conclude on Section 4.5 with some perspectives as well.

4.2 Related work

As mentioned in [82], one of the most challenging computations for the reputation of a given service is to track the indirect reputation. Indirect reputation refers to the situation where some services are either credited or penalized, because the customer facing provider is not the real provider.

Recommender mechanisms [17] rank the products or services based on feedback received after a series of recommendations and successful transactions. The *rating* is subject to incomplete, fictitious feedback, volume of transactions for a given product or provider, and confidence in feedback. Based on the ratings, the recommender computes its own *ranking* per product, defining the reputation (r) of a service/product. A computational mechanism including user' confidence (c) and feedback expectation (e) was proposed in [82].

An attempt, rather static, of considering a static subjective evaluation of the quality of the voice service is described by the MOS (Mean Opinion Score). The MOS is an arithmetic value ranging between 1 and 5, expressing individual perception [83]. However, MOS apply strictly to voice-related services, on an individual basis. The metrics are purely technical and related to codec use, packet loss, packet reorder, packet errors, and jitter. Another standard for evaluating the speech QoE, also considering technical metrics, is captured by PESQ algorithms [84].

A dynamic approach for customer input is presented by byClick system [85][86], where there is an on-line click-counting on the number of service accesses. No customer behavior, subscription status, or feedback patterns are considered.

However, in the current approaches, no correlation with the frequency of users' report and transaction peaks, as well as with the users' report patterns were considered.

4.3 Dynamic Feedback

The mapping QoS~QoE principally involves SLA' metrics. In our approach, we also introduce temporal and ethical metrics to quantify more accurately the customer feedback. Additionally, long term and short terms feedback patterns are identified, including spikes feedback.

We consider the basic introduced in [82], where by user expectation we mean the probability of having the user leave feedback after a service was delivered. The credibility refers to the user's ability to give a trusted rating. Usually both, expectation and credibility are expressed as percentage. The new mechanism proposed includes the status of the user and the feedback history.

Finally, we derive feedback-based policies for service reputation updates, by considering these metrics, based on extreme behaviors of customers in terms of feedback, e. g., feedback too late, too quick, too frequent, too rare, etc.

4.3.1 Dual reputation update architecture

Two views on reputation must be correlated for a given service/product, i.e., the provider view and the customer view. On the provider view, the perception of the service reputation, $r_{expected}$, represents the variation of several metrics, as the volume of sales, the number of new customers or lost customers in a given period. From the customer side, $r_{feedback}$ gathers customer perception on the reputation of a service (see Section 4.2.2).

Therefore, we propose a dual architecture to correlate and synchronize the two views. From the uniformity reasons, the update heuristics will follow a similar computation approach for both views, implemented by specialized engines.

Figure 4.1 depicts the main architecture and decisional and computation engines.

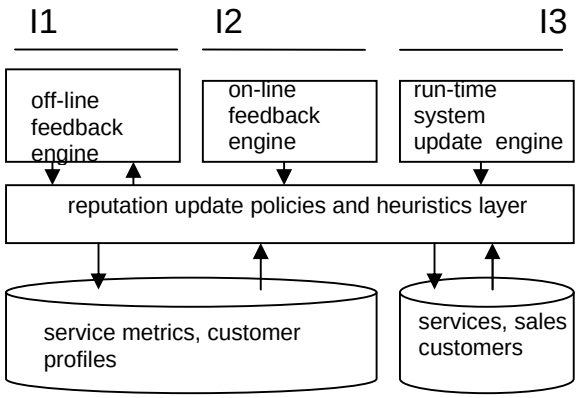


Figure 4.1. Dual reputation update architecture

The architecture considers two customer-facing interfaces (I1 and I2) handling the off-line (by_request, or at_will), and on-line (at_will), respectively, reputation updates.

I3 is considering the provider-facing reputation updates. Three appropriate *off-line, on-line*, and *run-time* engines deal with the updates, by receiving and computing them via I1, I2, and I3, respectively. In the current chapter, we only focus on aspects related to data collected via I1 and I3. For data collected via I3, an interesting implementation, nor related to our model, is presented by byClick [85][86]. While *dynamic* is different, the dual architecture can also support this approach. Hereafter, we will only refer to the off-line and run-time engines.

The *reputation update polices and heuristics layer* implements mechanisms to synchronize the two views, and to correlate the newly computed values. For example, a specific function is to trigger an update of $r_{feedback}$, when the, $r_{expected}$, has an unexpected variation, or when its variation trespass some thresholds. A concrete case could be when the volume of sales increases dramatically, with no appropriate variation of $r_{feedback}$. Appropriate heuristics will be presented in the next sections.

It is assumed that the customer and service records follow the model presented in [82] and enhanced in this proposal.

Without losing the generality, but for simplicity of the computation, we adopt the same formula, i.e., (4.1), as a core mechanism for reputation update engines; only the metrics will be specific to each view. Let us assume that a given service has a starting reputation r_0, on both views. We use the formula (4.2) for computing an updating value of the reputation:

$$r_{current} = r_0 \prod (1 + \lambda_i) \qquad (4.2)$$

where: $\{ \lambda_i = k_i \times w_i \quad i = 1\ldots..M\}$

M: the number of considered metrics

i: a given i-th metric [*i* belongs to N]

k_i: basic normalized update due to the variation of the i-th metric [k_i belongs to R]

w_i: weight factor associated with the i-th metric [w_i belongs to R]

The *off-line* and *run-time* engines compute the $r_{feedback}$ and $r_{expected}$, respectively, using (4.2) and appropriate heuristics for adopting λ_i. In the next section, we introduce a customer dynamic model and show how the λ_i are computed.

4.3.2 History metrics

We recall the main concepts of the enhanced recommender model [82], which we consider as the basis for dynamic feedback metrics:

s <r>: each service <s> has an associated reputation <r>

P_i<s,r_i>: each provider offers a service with its associated reputation

P_j<s,r_j>: another provider can offer the same service with a different associated reputation

u <e, c>: a user has a credibility and confidence metrics associated with

Note: for simplicity, we consider that <e, c> are the same for any service.

In evaluating the customer feedback, we consider individual metrics and metrics for a class of subscribers. In both cases, the feedback mode, i.e., 'by_request' or 'at_will' helps to differentiate between different extreme feedbacks.

feedbackMode::= {by_request, at_will}

1) Individual metrics

For individual metrics, ‘subscription seniority’, ‘feedback timing’, and the ‘satisfaction’ are relevant.

Seniority in profile::= {long term, regular, new}

FeedbackTiming::= {quick, regular, late }

SatisfactionDegree::= { x% | x = 0 - 100}

For policy-specification, we define satisfaction by metrics

satisfaction = satisfied, when $x > \beta_1$

= regular, when $\beta_2 \leq x \leq \beta_1$

= dissatisfied, when $x < \beta_2$

seniority = long term, when term $> \tau_1$

= regular, when $\tau_2 \leq$ term $\leq \tau_1$

= new, when term $< \tau_2$

feedback = quick, when $t < t_2$

= regular, when $t_2 \leq t \leq t_1$

= late, when $t > t_1$

With the above definitions, we assume that the architecture handles the seniority of the subscribers and the timestamps of their feedbacks after a service was consumed.

The following patterns of interest can be identified for each seniority profile:

a. [<quick><satisfied>]

b. [<quick><dissatisfied>]

c. [<late><satisfied>]

d. [<late><dissatisfied>]

The profile metrics quantified as ‘regular’ do not alter the computation of the reputation.

With the new metrics, a user is characterized by

- expectation
- credibility
- seniority

and a ‘per service’ feedback pattern. The feedback patterns comprise:

- feedback timing
- satisfaction degree
- feedback dynamics (#satisfied, #dissatisfied, repetitive replies, observation period)

There is a calibration phase for each system, where the appropriate values are tried and settled for the thresholds. For example, the following steps are considered by a calibration procedure for the feedback timing:

(1) An ‘average’ reply time is observed and recorded for both feedback modes for a given service.

(2) After the calibration period, the customer reaction is observed for that service, called ‘average’.

(3) A policy can be defined by heuristics, as follows:

```
START
IF feedback mode = at_will
    IF 'feedback timing' is 'three times' than the
       'average'
           THEN feedback = late
IFNOT (feedback mode = at_request)
     IF 'feedback timing' is 'twice' than 'average'
           THEN feedback = late
END
```

Heuristic #1. Settling the feedback values

In a similar way, and based on calibration, policies for settling each threshold can be defined.

2) Metrics for classes of subscribers

For a class of subscribers to the same service, we propose feedback metrics capturing the community behavior. In the case of a community, the individual profiles are aggregated. In order to capture the dynamicity of the feedback, we introduce a few feedback metrics describing a pattern structure. In a given observation period (Δ), we define the number of repetitive replies (m) and the number of satisfactions (ni+) and dissatisfactions (ni-), as well as n- = max {ni- | i = 1…} and n+ = min {ni+ \ i = 1….}. For example, in Figure 4.2, on the top, m = 2, n1+ = 4, n2+ = 5, and n+ = 4. In the third basic pattern, when both satisfactions and dissatisfactions are present, a pair (n+, n-) is attached to it.

Based on a series of observations periods, a profiling system is able to classify customers and have a coarse granularity on the feedbacks.

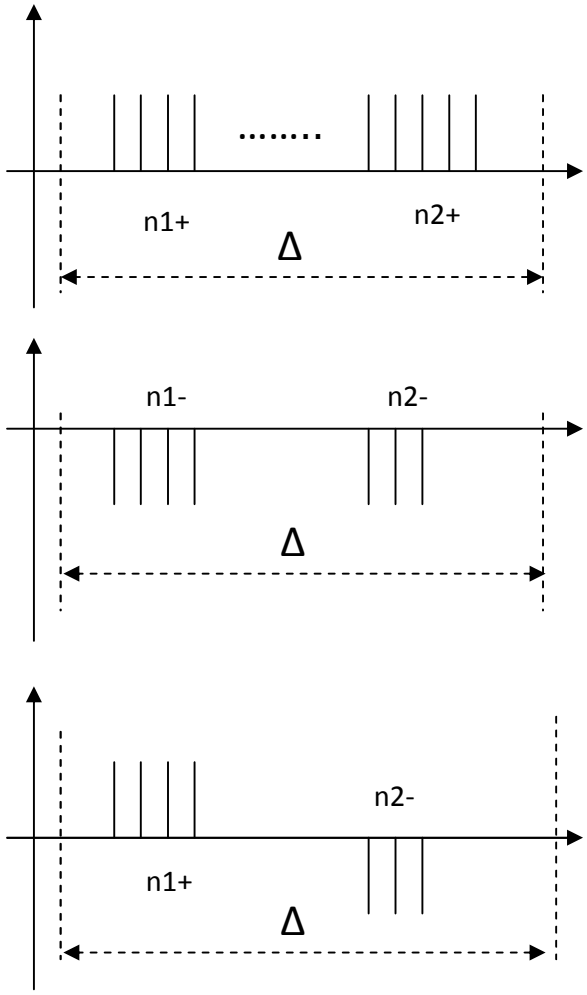


Figure 4.2. Basic feedback patterns

With these metrics, a reputation engine can trigger appropriate reputation update mechanisms. Definitively, they can be combined with the user history metrics to build more complex (and more accurate) updates.

4.3.3 Dynamics on service subscriptions

Service reputation is a main metric to justify the existence of that service, new investment in developing new features of the service, and new marketing activities to promote a given service.

In our model, we consider a few metrics that portray the dynamics of service sale, e.g., the volume of transactions, the number of new customers in a given period, and the number of lost customers in that period. For the last metric, we also consider the seniority, therefore distinguishing between long term and new customers.

Therefore, apart its features from the quality point of view, from the reputation perspective, a service is described by:

- r: reputation
- vol: variation in transaction volume [± %]
- new: new customers [%]
- lost_new: lost new customers [%]
- lost_long: lost long term customers [%]

(% is considered versus the numbers at the beginning of the observation period; usually every update represents the start of a new period)

A revision of the reputation of a service is always based on the above metrics.

An example of using heuristics for updating the reputation based on the number of transactions can be:

START
IF vol = - 10%
    THEN $r_{real} = (1- 0.1) \times r_{current}$
ELSE
    $r_{real} = r_{current}$
END

Heuristic #2. Updating the reputation versus variations of transactions

4.3.4 Conclusion on the reputation update model

We presented a model for updating the reputation of a service considering the user profile, its dynamic feedback, on the one side, and the dynamics of service subscriptions.

In general, the reputation updates is triggered by a significant variation of one of the service subscription dynamics. This moment defines the origin of an updating period. It is assumed that a recommender system records other customer feedback information that is considered when updating the reputation.

In the following section, we present some basic heuristics to update service reputation, considering the metrics described above.

4.4 Dynamic Reputation Updating

There are several classes of updates, based on what metrics are used.

4.4.1 Satisfaction and feedback based policies

The simplest way of updating the reputation is considering the first four patterns

a. [<quick><satisfied>]

b. [<quick><dissatisfied>]

c. [<late><satisfied>]

d. [<late><dissatisfied>]

Following Policy #1, we associate with (a) and (b) a correction θ_1 and with (c) and (d) a correction θ_2, with $\theta_2 < \theta_1$, when the feedback_mode is 'at_will' and with $\theta_4 < \theta_3$, respectively, when the feedback_mode is 'by_request', with $\theta_4 < \theta_3 < \theta_2 < \theta_1$. The subjective justification of these values is given by the customer attitude in terms of promptness of reactions and their qualification.

The correction, in this case, is expressed by the following policy [Policy#1]

START
IF feedback_mode = *at_will*
  IF feedback = *quick*
    IF satisfaction = *satisfied*
      THEN $r_{real} = (1+ \theta_1) \times r_{current}$
    IFNOT (satisfaction = *dissatisfied*)
      THEN $r_{real} = (1- \theta_1) \times r_{current}$
  IFNOT (feedback = *late*)
    IF satisfaction = *satisfied*
      THEN $r_{real} = (1+ \theta_2) \times r_{current}$
    IFNOT (satisfaction = *dissatisfied*)
      THEN $r_{real} = (1- \theta_2) \times r_{current}$
IFNOT (feedback_mode = *at_request*)
  IF feedback = *quick*
    IF satisfaction = *satisfied*
      THEN $r_{real} = (1+ \theta_3) \times r_{current}$
    IFNOT (satisfaction = *dissatisfied*)
      THEN $r_{real} = (1- \theta_3) \times r_{current}$
IFNOT (feedback = *late*)
    IF satisfaction = *satisfied*
      THEN $r_{real} = (1+ \theta_4) \times r_{current}$
    IFNOT (satisfaction = *dissatisfied*)
      THEN $r_{real} = (1- \theta_4) \times r_{current}$
END

Policy #1: Reputation updates #1

The values of all weights are done by validated calibration. For example, if the orders of a given service do not increase (or decrease), it means that the reputation is too high. Therefore, reputation is always 'in question', when the transactions for a service vary, or the service orders show a quick increase or decrease (in terms of volume, and in terms of new customers).

4.4.2 Satisfaction, feedback, and seniority based policies

Let us assume that a mechanism is in place for complying with Policy #1; additionally, the seniority must be considered. There are two strategies used in our model to update the reputation: (i) an optimistic one, and a (ii) pessimistic one.

Let us assume we have the following situation:

vol = +15%

new = 10%

lost_new = 2%

lost_long = 1%

In an optimistic strategy, would credit the 'vol' and 'new', while downplaying the 'lost_new' and 'lost_long'. Following the same approach of correcting by fraction representing the percentage, i.e., 10% is a correction of 0,1, we have the following heuristic:

START
IF vol = +15%
new = 10%
lost_new = 2%
lost_long = 1%
THEN r_{real} = (1+ 0,15) (1 + 0,1) (1 – 0,02) (1- 0,01) x $r_{Policy\#1}$
END

Heuristic #3. Considering variations in transactions and subscribers (optimistic)

In a pessimistic approach, losing new subscribers or long term subscribers is an indication of service degradation from the quality point of view, of a violation of the SLA with a significant number of subscribers, or simply that the service was not upgraded at the expected standard.

In this case, a multiplicity factor can be used to consider the loss, e.g., k1 for loosing new subscribers and k2 for loosing long term subscribers.

START
IF vol = +15%
new = 10%
lost_new = 2%
lost_long = 1%
THEN r_{real} = (1+ 0,15) (1 + 0,1) (1 – k_1x0,02) (1- k_2x0,01) x
$r_{Policy\#1}$
END

Heuristic #4: Considering variations in transactions and subscribers (pessimistic)

Calibrating the values for k1 and k2 in this case follows also a given heuristic, as expressed below:

	1T	2T	3T
k1	3	2	1
k2	6	6	6

Heuristic #5: Multiplicity correction factors

In Heuristic #5, an example of selecting the multiplicity correction factors is presented. Assume that the unsubscribe event occurs in 1T, 2T or 3T time units for the enrolment, the example gives more weight to the loss of long term customers ($3T < \tau_2$ to correctly evaluate the 'new').

Note1: In the model presented above we considered no difference between the types of service, assuming that the QoS, from the provider perspective was delivered according to the SLA.

Note2: In the heuristics and the metrics presented above, we didn't consider any emotional feedback that might influence the feedback (such as accompanying gifts, bonuses, or penalties for QoS violations), nor particular interests of a customer in a service provider, such as stocks.

4.4.3 Reputation update considering feedback patterns

A fine grain reputation update considers the feedback patterns presented in Figure 4.1.While only one pattern can be considered to update the reputation, Heuristic #6 considers all three patterns (see Figure 4.2).

START
IF (Δ, m, n^+) [$\Delta > \tau_1$]
THEN $r_{real} = (1+ m^+xn^+/100) \times r_{current}$
IF (Δ, m, n^-) [$\Delta > \tau_1$]
THEN $r_{real} = (1- m^-xn^-/100) \times r_{current}$
IF ((Δ, m, n^+, n^-) [$\Delta > \tau_1$]
THEN $r_{real} = (1- m^-xn^-/100)\ x(1+ m^+xn^+/100)x\ r_{current}$
END

Heuristic #6. Reputation updated considering the feedback patterns

4.5 Reputation Updating Policies and Heuristics

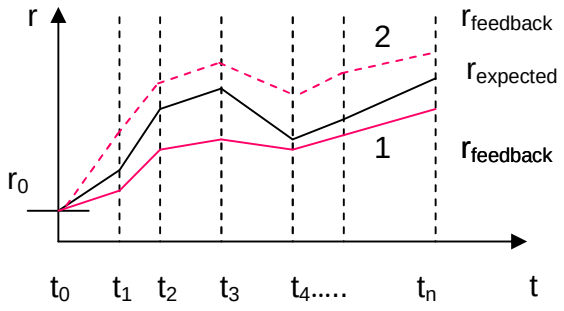
By their own nature, from both views (customer, provider), the reputation values on each view is a list, similar with time series, with

$r_{feedback}$ = {r1, r2, r3, …….}, and (4.3)

$r_{expected}$ = {r'1, r'2, r'3, ……}

at {t1, t2, t3, …..}

The reputation values can have the following position, as shown in Figure 4.3.



a)

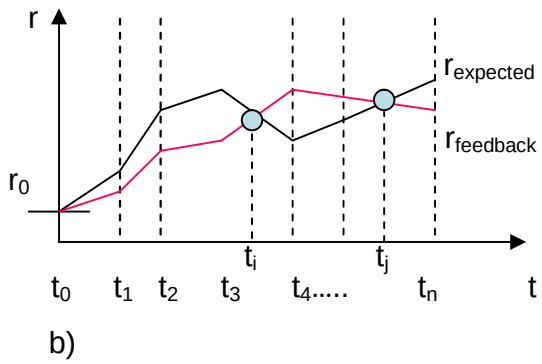


Figure 4.3. Relative position of reputation values

While computing the real reputation on both views, and considering the dynamics of customer feedback, we can observe a *channel trend* (Figure 4.3a) or local anomalies (Figure 4.3b). Formally, there are several cases for defining updating heuristics.

A chann*el trend* is considered when $| r_{expected} - r_{feedback}| < \varepsilon$ and an *anomaly* occurs when $| r_{expected} - r_{feedback}| > \delta$, where $\varepsilon << \delta$, for all t_i (ε and δ are thresholds that are established per services).

The model allows implementing five heuristics for synchronizing the feedback and expected reputation (see (4.3)). This allow to adjust the market from some actions; with no lost generality, we only consider three potential actions, i.e., 'increase/decrease the storage order', 'increase/decrease the price', and 'increase/decrease the offered QoS'. These actions are main contributors for the service costs.

1) Synchronization

In this both expected and feedback reputations are in synchronization with small variations:

IF
$| r_{expected} - r_{feedback}| < \varepsilon$
AND
$r_i > r'_i$ for all i
OR
$r_i < r'_i$ for all i
THEN "keep same storage order"
AND
"keep same prices"
AND
"keep same QoS/SLA agreements)
END

Heuristic #7. Regular computation

No particular actions are triggered, but regular reputation computation by off-line and run-time engines.

2) Pessimistic anomaly (from the provider)

In this case, the feedback reputation is much higher than the expected reputation; it is a situation to increase the objective function (benefits).

IF
$| r_{expected} - r_{feedback}| > \delta$
AND
$r_i > r'_i$ for all i
THEN
IF 'no QoS violation"
THEN "increase storage order"
ELSE
"offer lower QoS/SLA provider metrics"
(fewer costs)
END

Heuristic #8. Pessimistic policy

3) Optimistic anomaly (from the provider)

In this case, the expected reputation is much higher than the feedback reputation; it is a situation to decrease the objective function (benefits).

IF
$| r_{expected} - r_{feedback}| > \delta$
AND
$r_i < r'_i$ for all i
THEN
IF 'no QoS violation"
THEN "reduce storage order"
OR "reduce price"
ELSE
"offer better QoS/SLA provider metrics"
(more costs, to attract customers)
END

Heuristics #9. Optimistic policy

4) Under estimation (by the provider)

This case refers to the situation where the expected reputation decreases, but the feedback reputation increases.

IF

$| r_{expected} - r_{feedback}| < \varepsilon$
AND $r_i = r'_i$ for a given I [see t_i in Fig. 4.3b]
AND $r_{i-1} < r'_{i-1}$

AND $r_{i+1} > r'_{i+1}$
THEN
(expectation decreases, customer satisfaction increases)
IF 'no QoS violation"
THEN "increase storage order"
OR "increase price"
ELSE
"offer lower QoS/SLA provider metrics"
(less costs))
END

Heuristics #10. Under estimation policy

5) Over estimation (by the provider)

This case refers to the situation where the expected reputation increases, but the feedback reputation decreases.

IF
$| r_{expected} - r_{feedback}| < \varepsilon$
AND $r_j = r'_j$ for a given I [see t_j in Fig. 4.3b]
AND $r_{j-1} > r'_{j-1}$
AND $r_{j+1} < r'_{j+1}$
THEN
(expectation increases, customer satisfaction decreases)
IF 'no QoS violation"
THEN "decrease storage order"
OR "decrease price"
ELSE
"offer better QoS/SLA provider metrics"
(more costs)
END

Heuristics #11. Over estimation policy

4.6 Conclusion

In this chapter, we presented an approach for modeling the dynamic user feedback, when computing the reputation of a service. We presented specific metrics describing the customer feedback, on the one side, and the variations in service transactions and customer subscriptions, on the other side. We showed that the metrics can be used to specify different heuristics leading to a more accurate reputation. We proposed a dual architecture to compute service reputation from both provider and customer views, and build a dynamic customer feedback model and appropriate heuristics to adapt the service delivery activities, accordingly. We have shown how this approach can be used to trigger appropriate actions to optimize service delivery under agreed QoS/SLA metrics.

The next chapter will focus on the computation of a service reputation considering service similarities and validation of the model in this case, as well as a generalized formalism to capture the model. We will present a more formal definition of service/provider/feature similarity, and the stability of the reputation accuracy over a longer period. This might lead to the reputation predictions; specialized metrics for assessing the accuracy of predictions in the light of indirect delivery are challenging but seen as very helpful in web-service driven environment.

Next challenging issues are related to the computation of a service reputation considering service similarities and validation of the model in this case, as well as a generalized formalism to capture the model.

CHAPTER 5

Online Service Similarities and Reputation-based Selection

Summary

Chapter 5 presents adapted approaches to select services based on distance and similarity, and introduces similarity taxonomy to better tune various kinds of service invocation under specific constraints, such as relaxation, type of similarity, context, and service ranking. Selection is also based on the feedback from the user.

In order to evaluate service similarity, we need to have certain service features that can be compared. These features are compared taking into consideration context and also their composition. Two features, a_{1k} and b_{2z}, of two different services, might not be semantically matching, but they might be equivalent if feature a_{1k} is composed with feature a_{1p}, making feature b_{2z} a subset of the composition. Also, different users may consider the same features mandatory or not mandatory. For this purpose, we introduce the notions of primary service features and secondary service features.

The proposed model is used for building a selection algorithm that allows variations on service invocation.

5.1 Introduction

The large spectrum of user behaviors (and, in general, the variety of needed services) leads to the need of similarity-based matching, when a given service is required. Traditionally, the notions QoS (Quality of Service), and QoE (Quality of Experience) deals with these aspects. However, the perfect matching and the approximate matching depend on a large number of factors. For example, if we consider Web Services dedicated to weather forecast, location, month/day/year, parameters (rain, wind, temperature, and pressure) can be appropriate parameters when inquiring. Definitively, there are several forecast services, and the experience of a particular user might differ from one forecast service to another. Some provide information that is more accurate than others (i.e., data is more frequently updated), history is better preserved by particular services, via backward search, e.g., Weather Underground, etc. A similar problem is observed when choosing and downloading a particular piece of software, when inquiring for a specialized on-line book shop, or when looking for a service providing the most updated world-wide information. Finally, some services offer a friendlier interface for searching, ordering, and getting delivered a particular need (i.e., personalized interface, myAccount, etc.).

There are meta-services, providing the service at choice. Such examples are those for buying flight tickets, where the cheapest, the quicker, or other selection criteria are used for service selection. Other meta-services are for selecting the most appropriate

software to download, or for booking a hotel. In most of the cases mentioned above, one criterion is usually considered to select from an existing service pool.

Two phases involve service features, (i) service discovery (locating) and (ii) service selection (in the case of a set of services, relatively satisfying the needs with similar degrees of satisfaction). Both phases require special mechanisms to assess service similarity. Meta-services have a restrained number of known services, that are well localized and whose parameters are also limited. Because of this, the selection appears to be less complex. With a well known service and limited criteria (usually one search/ selection criterion), similarity is relatively easy to be determined.

The above considerations are no longer valid for a large spectrum of properties a service might expose in order to satisfy a given service request. To satisfy a request, service similarity plays an important role for timely identification and delivery, and for an optimal (maximal) customer (invoker) satisfaction. Customer satisfaction is expressed by QoE, on-line feedback, service ranking, and manifested by variations of QoS to keep service costs and satisfaction in synchrony.

This chapter deals with service similarity and proposes a adaptive similarity taxonomy and mechanisms to handle service discovery and service selection considering service specification, end-user (requester) perception, and service reputation. In Section 5.1, existing approaches for service similarities are presented. Section 5.2 introduces a context-based similarity model, including distance and similarity metrics, a similarity taxonomy, and other facilities to consider service ranking and feature relaxations. Section 5.3 presents an algorithm to compute a minimum set of existing services satisfying a

given query, following the newly introduced model. Section 5.4 presents a global analysis of the approach presented in this chapter. Section 5.5 concludes on the approach and presents further developments.

5.2 Related Work

Finding similar services (approximate but satisfactory matching) is somehow similar to (i) text matching, (ii) schema matching, or (iii) software-component marching. For some text matching solutions (information retrieval) mechanisms based on term frequency are used [87][88]. In schema matching, special techniques are using semantics of the schemas to suggest schema matching [89]. Mainly, linguistic and structural analyses, as well as domain knowledge, are methods to handle schema matching. When expanding to software component matching [90] (considerably used in software reuse) component signature and program behavior (usually formally defined) are considered; in this case, data types and post-conditions should be considered for matching. However, these techniques are not suitable for Web Services [91], as data types and post-conditions are not available. Usually, such a service has a name and text description in UDDI **(**Universal Description, Discovery, and Integration) registry, operation decryptions, and input/output descriptions; the last two are usually specified in WSDL (Web Service Description Language).

Dong *et al.* [91] proposed criteria for associating similar terms. They introduced the cohesion/correlation score, as a measure of how tight two terms are. However, they do not consider particular characteristics of a term. They applied the score

only to Web Services. We start from the idea that services similarity has a meaning only between services than can be context-oriented and belong to a cluster (e.g., invoking a service gives a list of similar operations with similar results). Other approaches consider both diversity and similarity at the same time, having the distance as a metric [92]. We adopt these metric (see Section 5.3) and adapt them to the service similarity computation.

In fact, specific to each service, there are particular service parameters that are agreed upon between a provider and a subscriber, commonly settled by the SLA (Service Level Agreement). On the provider side, the SLA parameters are used for technical audit and litigations (leading to penalties or bonuses towards a given user or class of users). Specific on-line and off-line measuring mechanisms for SLA metrics and specialized audit techniques have been proposed. On the consumer side, the subscribers' satisfaction is gathered and mapped to the audit results to validate a given service, to detect flaws in delivering a service, and to ultimately build a view on service reputation. In general, a record is handled per service or per products, with respect to a given subscriber or a class of subscribers. Feedback can be used to enforce service similarity.

In this chapter, we expand the cluster-based similarity to service similarity and introduce similarity taxonomy, where the service consumer has a weight in deciding service similarity. The idea is to establish service ranking (and reputation) inside a given cluster, and define similarity considering service-provider and service-user feedback.

5.3 A Context-based Similarity Model

Then main idea of our approach is (i) having well defined service clusters, (ii) compute the distance between service feature, (iii) evaluate service similarity, based on service features, (iv) consider user-, service,- and producer-based similarity reflected by the appropriate reputations, and (v) evaluate how interchangeable two services are. When a service query is issued, the algorithm we propose selects the most appropriate service, considering both distance and similarity between services.

5.3.1 Identifying clusters of similar services

Expanding what was mentioned in [91], service cohesion of a service cluster must be strong (best potential to be similar), while correlation between two service clusters should be weak (service independence). We say that service s_1 is similar with s_2, and note $s_1 \sim s_2$, if the similarity confidence is greater than a given threshold δ. In a cluster S with ||S||, where ||x|| is the cardinality of x, we redefine cohesion and correlation as follows:

$$Cohe_S = \{(s_i, s_j) \mid s_i \sim s_j \ (\sim_{thres} > \delta)\} \ / \ (||S|| \ x \ (||S|| - 1)) \quad (5.1)$$

and

$$Correl_{S,S'} = (A \ (S, S') + A \ (S', S)) \ / \ 2 \ x \ ||S|| \ x \ ||S'||, \quad (5.2)$$

where

$$A \ (S, S') = || \ \{s_i, s_j \mid s_i \in S, s_j \in S' \ si \sim sj \mid \sim_{thres} > \delta\}|| \quad (5.3)$$

with $\sim_{score} = Cohe_S / Correl_{SS'}$ (5.4)

defining the similarity score.

We notice that $\sim_{score}$ defines similarity classes based on the preexisting service clusters. To enhance the similarity score, clusters aggregation and clusters split operations are possible. Conditions and assessments for doing these are presented in [91].

5.3.2 Distance metrics for service similarity

Let us assume that a service s has n features (usually called data-points, as they are expressed by concrete values in an n dimensional space). The following distance methods are adapted for comparing services:

(a) Service Euclidian distance between two services in the n dimensional space

$d(s1,s2) = 1/n \Sigma (a1i – b2i) **2$, for all i = 1…n (5.5)

where ai, bi are service features.

(b) Service city-block distance

$d(s1,s2) = 1/n \Sigma |a1i – b2i|$, for all i = 1…n (5.6)

(c) Service Pearson correlation coefficient

$r(s1,s2) = 1/n \Sigma ((a1i – a)/\sigma a) \times ((b2i – b)/\sigma b)$, (5.7)

where a and b are the sample mean of ai and bi respectively, and σa and σb are the sample standard deviation of ai and bi.

The service Pearson distance is defined as

$$d(s1,s2) = 1 - r(s1,s2) \quad (5.8)$$

(d) Service Cosine similarity

$$d(s1,s2) = \cos(\theta) = (s1 \bullet s2) / (|s1| |s2|) \quad (5.9)$$

where ● is the vector product of s1 ans s2.

By selecting a service distance metric, a clustering algorithm computes the distance matrix between two services. Mostly, (a) and (b) of the above are satisfying the triangle inequality, as true metrics.

5.3.3 Classes of similarities

In order to select the most appropriate service, we introduce producer-based similarity ($\sim_{prod}$), recommender-based similarity ($\sim_{recc}$), and user-based similarity ($\sim_{user}$). Producer similarity is based on the expectation, recommender's similarity is statistics-based, and user similarity is based on user feedback. In this taxonomy, $s_1 \sim_{prod} \{s_2, s_3, \ldots\}$ define a cluster of similar services, as defined by the producer.

To refine service similarity, we introduce the notions of *primary service features* and *secondary service features*, as shown in Figure 5.1,

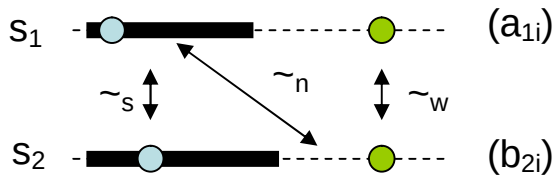


Figure 5.1. Similarity classes.

where the bold items represent primary service features (A_1 set), and the dashed items represent secondary service features (A_2 set) (similar for s_2)

We introduce strong, weak, and normal similarity, represented by $\sim_s$, $\sim_w$, and $\sim_n$, respectively.

Therefore, $(s_1 \sim s_2)$ =

$= \sim_s$, iff all $a_{1i} \in A_1$ and $b_{2i} \in B_1$

$= \sim_w$, iff all $a_{1i} \in A_2$ and $b_{2i} \in B_2$

$= \sim_n$, iff there are $a_{1i} \in A_1$ and $b_{2i} \in B_2$ or

there are $a_{1i} \in A_2$ and $b_{2i} \in B_1$ (5.10)

Similarity composition allows to capture all possible combinations, e.g,, $\sim_{prod/s}$ represents a strong similarity defined by the producer, based on the primary service features.

A refinement of feature-based similarity can be expressed when service features do not show a direct semantic matching, but feature composition might lead to such a match. Considering a subset of a service feature for a given service equivalent with a feature for a service for which the similarity is computed, we introduce feature composition-based similarity, as shown in Figure 5.2.

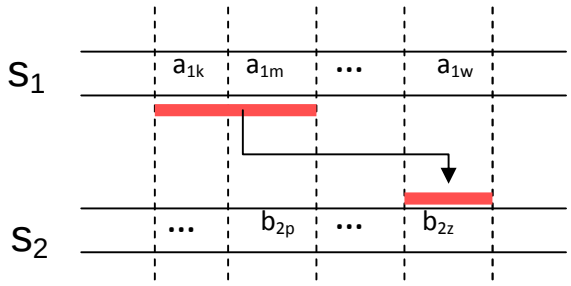


Figure 5.2. Feature composition-based similarity

$$s_1 \sim_{a1k,a1m\,/\,b2z} s_2 \tag{5.11}$$

with the semantic that the values of a_{1k} and a_{1m} composed are similar to the values of b_{2z}. Composition might be any arithmetic or Boolean operator, according to the nature of the features, e.g., if sets, then 'U' (union), if values, then '+' (addition), etc. If type, and a_{1k}:T1 and a_{1m}:T2, and b_{2z}:T3, then, then T3 is a subtype of either T1 or T2.

Combination between $\sim_s$, $\sim_w$, and $\sim_n$, and feature composition-based similarity can be applied following (5.10).

5.3.4 Updating Similarity

When evaluating service similarities, perfect match of service features is desired, but rarely found, due to some continuous values of the features. For example, looking for a service offering the weather temperature with an accuracy of 0.1^oF is not feasible. A query on what month the temperature is 67.3F might have no match; but, for a given location, a query for what month shows [75-80] ^oF might be answered by April or May, if

a Mediterranean area. We identify two possible relaxations when performing the matching.

5.3.4.1 Context-based Feature Migration

In time, and based on business models or customer feedback, some primary features become secondary, and vice-versa. Even more, at the same time, in different contexts, a feature can belong to either primary or secondary feature sets.

Let $C = \{c_i\}$ a set of contexts and

$$s1 ::= (A_1 U A_2)_{context = c1}, \text{ with } A_1 \cap A_2 = \phi$$

$$s1 ::= (A'_1 U A'_2)_{context = c2}, \text{ with } A'_1 \cap A'_2 = \phi \quad (5.12)$$

then, the following is possible:

$$s_1 \sim_{context = c1} s_2$$

$$s_1 \sim_{context = c2} s_3 \quad (5.13)$$

5.3.4.2 Feature Relaxation-based Similarity

Service features are not always perfectly matching (so goes for query matching, as well). Most of the time, the exact matching is not mandatory, or, at least, the query can explicitly mention an acceptable variation. Usually, this is expressed as a constraint associated with a given service feature. For example, requiring a book delivery service, might have as a condition, *delivery costs* < *threshold*. In other cases, if a service feature has a numeric value a variation of a_{1i} (usually symmetric, but not necessarily) of +/- α_{1i} is

allowed. As a result, the similarity metrics presented in 5.2.2 can be relaxed. The same relaxation can be applied for similarity on data type/subtype, for similarity concerning the set of interface operations, or similarity concerning variations of an algorithm implementation.

For s_1 $[a_1, a_2, a_3, \ldots a_n]$

s_2 $[b_1, b_2, b_3, \ldots b_k]$

Let us assume that a few service features a_i are associated with constraints. These constraints may be expressed as follows:

a_i > expression/threshold

a_i < expression/threshold

$a_i \in [x, y]$ (*belongs to*, as an interval)

$ai \in \{x, y\}$ (*belongs to*, as a set)

For a service selection, all expressions must be returned TRUE.

A query for a service can be represented by:

Q (s, similarity type, context, $\{(a_i, \text{constraint}_i)\}$)

We express this as

$s_1 \sim_{\text{constraint}} s_2$ <=> bi satisfies a_i, and all constraint_i are evaluated TRUE, for ALL *i* mentioned in the Q

For example, when a query (with explicit relaxation of +/- 2ms) targets a service with a response delay of 10ms, any service offering a delay within [8ms, 12ms] is a desired matching. With no explicit relaxation delay, 10ms is mandatory. In this case,

$$s1 \sim_{a1i +/- \alpha 1i} s2 <=> b_{2i} \in [a_{1i} - \alpha_{1i}, a_{1i} + \alpha_{1i}] \quad (5.14)$$

where a_{1i} and b_{2i} are the corresponding features of s_1 and s_2, respectively.

As a note, similarity with constraints increases the chance to have a matching to a given query, on the expense of additional computation.

A variation of this kind of similarity is when constraints are:

- Mandatory, for primary features (M)

- Optional (while desired), for secondary features (O)

For expressing these variations, a Q must be explicit on the categories of features

Q (s, similarity type, context, M:{(a_i, $constraint_i$)}, O:{(a_i, $constraint_i$)})

A response for the system should also contain the reference to the kind of feature/similarity, e.g.,

A response can be

$\{s_{1/M/nonO}, s_{3/M/O}, s_{8/M/nonO}\}$, or simply

$\{s_{1/M/nonO}, s_3, s8/M_{/nonO}\}$,

where *nonO* index represents the feedback of not all optional constraints were evaluated TRUE.

5.3.5 Recommender-based Similarity

Recommender mechanisms rank [82] the products or services based on feedback received after a series of recommendations and successful transactions. The *ranking* is subject to incomplete, fictitious feedback, volume of transactions for a given product or

provider, and confidence in feedback. Based on statistics, the recommender computes its own *ranking* per product, defining the reputation (r) of a service/product.

Considering a set of service clusters a recommender builds based on type of services/products,we define:

Cluster = $\{cluster_i\}$

with $s_1 \epsilon cluster_i$ and $s_2 \epsilon cluster_{i,}$ for a given service feature

$$s_1 \sim_{feature = ai} s_2 ::= |rank_{s1} - rank_{s2}| < \varepsilon_{ai} \quad (5.15)$$

In general,

$$s_1 \sim_{Uai} s_2 ::= \max \{|rank_{s1} - rank_{s2}|\} < \min \{\varepsilon_{ai}\} \quad (5.16)$$

5.3.6 Customer feedback reputation-based similarity

Based on customer individual metrics, context, and potential query with relaxation, a reputation is associated with a service/product. Heuristics for updating the reputation have been presented in [82][93]. In general, the following information is available:

s <r>: each service <s> has an associated reputation <r>

P_i<s,r_i>: each provider offers a service with its associated reputation

P_j<s,r_j>: another provider can offer the same service with a different associated reputation

u <e, c>: a user *u* has a credibility and confidence metrics associated with it

For simplicity, we consider that <e, c> are the same for any service.

For a given user, we define similarity in terms of r_s

$$s_1 \sim_{feedback} s_2 ::= |r_{s1} - r_{s2}| < \varepsilon_0 \text{ with } e > e_0 \text{ and } c > c_0 \quad (5.17)$$

In the following, the newly introduced model is used by an algorithm to identify the most suitable service to satisfy a query for a service.

5.4 Algorithm for Service Retrieval Using Similarity

We introduced a similarity model and classes of similarity that allow a user (invoker) to use a service in a given context, allowing or not precise relaxation for some service features, and under different types of similarity (strong, weak, normal). Distance metrics were also adopted for services, in order to cluster the most suitable services for a particular query, before computing the similarity.

Based on the model introduced in Section 5.2 and on the user model [93] and reputation [82], a query for a service *s* can be expressed as

Q (s, similarity type, context, with/without relaxation on $\{a_{1i}\}$)

The algorithm presented below illustrates the main steps to reach a service proposal that can be a set, a given service, or no service at all.

Algorithm for finding a requested service query **Q**, *based on similarity between potential satisfying services*

1: **begin**
2: **identify** the service cluster [see (5.4)]
3: **select** a distance metric [see (5.5)-(5.9)]

4: **calculate** distance between all s_i in the cluster
5: **select** a subset $\{s_k$ with min $\{d(s_i, s_j) < \varepsilon\}$
6: **if** Q with relaxation
7: **apply** (5.10) and (5.11) for all mentioned features
8: **if not**
9: **if** Q with context
10: **apply** (5.12) and (5.13)
11: **if not**
12: **compute** a subset {si} of the set found before step12
13: **select** {sl} from the subset of step 12, with rank $(s_l) > \delta_1$ and $r_{feedbakc} > \delta_2$ [see (5.16) and (5.17)]
14: **select** a subset for the subset of step 13
15: **return** the subset of step 14
16: **end**

Note that the output of step 15 might be an empty set, or a set having many recommended services complying to the query conditions.

The complexity of the algorithm is given mostly by the number of service features that can be considered with relaxations.

A variation of the algorithm was experimented with relaxation conditions for a set of contexts. The number of features with relaxation, the number of contexts, and the number of services into a cluster determine the performance of the algorithm.

Different experiments on the on-line Barnes & Nobles system (on-line bookstore) show a reasonable improvement on the precision the algorithm returns after running various numbers of queries and varying different conditions.

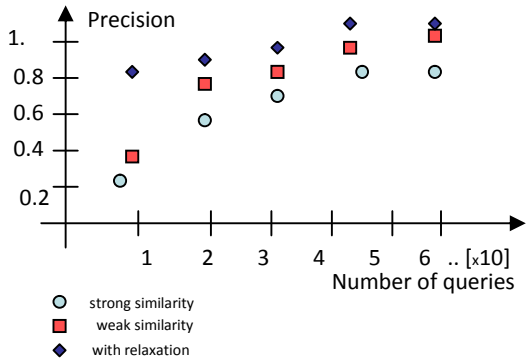


Figure 5.3. Precision of service returned to queries with different types of similarities

It is no surprise that a service satisfying a query with relaxation reaches faster and with a higher precision the query expectation.

Two performance improving procedures are possible: (i) Building a query cluster each query belongs to (as a set of a priori known services that might satisfy the mandatory features of a given query, and (ii) Based on the responses, it is interesting that the same similarity criteria used to identify similar services can be used to evaluate the 'satisfaction' of the returned services.

The first procedure needs a look up in the previous query inventory, and group them by context and user. Then, looking by context and user ID, a subset of services are derived. A Q is now answered by considering a particular service cluster, sensitively smaller than the entire set of services.

By running a few situations, with 1,000 services, 5,000 contexts, 8,000 users, and queries with 4 mandatory features and 5 optional features, no constraints, the following results were obtained.

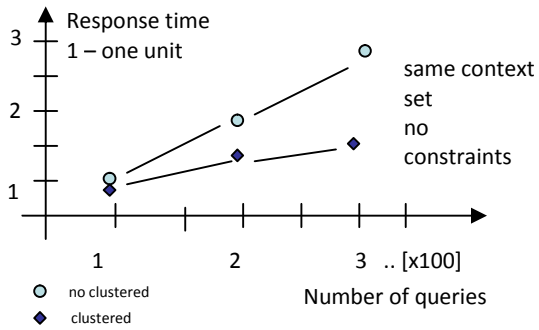


Figure 5.4 Response time versus number of queries in clustered and non clustered approaches

In the case where services are clustered (based on previous answers) the response time is dropping by almost half. However, a similarity (Q, A) must be also executed to evaluate how the recommended services satisfy a given query.

In terms of returned services, under the same setting, the results are presented in Figure 5.5.

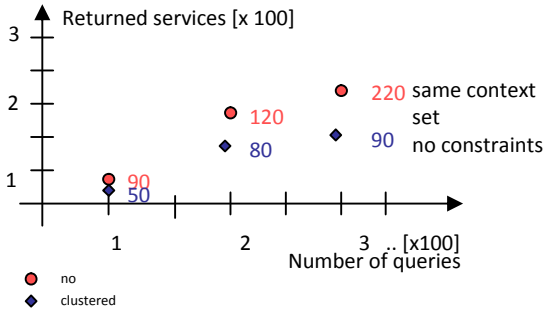


Figure 5.5. Returned services versus number of queries in clustered and non clustered approaches

A sensitive drop in returned services is observed in the clustered mode; however, it seems to be saturation with the number of queries increasing. This can be caused by the lack on context differentiation, or by the similarities of the queries. On the latter part, more experiments are needed.

The second procedure is used to apply similarity (Q, A) for each service in the retuned set, and select that service that has the max similarity. The procedure is simple, but requires to be applied, in turn, to all returned services. To simplify, one may select to run the similarity check only for the primary features. In this case, with the same settings for the experiment, the computation time is reduced by two thirds. This is due to the fact that similarity with constraints requires additional computation for each feature to validate that the constraint is TRUE.

To substantially reduce the computation time, and the cardinality of the returned services, a condition on service reputation reduced the time and the number of returned services.

5.5 Global analysis

As a global analysis on the approach presented in this chapter, we analyze a few variants:

(i) How to improve the precision when forming the clusters

(ii) Provide the precision support for similarity with constraints

(iii) Analyze the impact of features aggregation when similarity is applied

(i)

There are two phases where the service clusters are formed, i.e., (i.1), when the services are originally instantiated, by contexts, usually done by service producers, and (i.2), when services are grouped by past experiences as a result of returned answers for customer queries.

In (i.1), the most used criterion is service reputation in a given context; therefore, the framework and the mechanism introduced in the thesis must be used.

For (i.2), a solution based on clustering queries and then associating for each query cluster a potential service cluster(s) is optimizing the search. When considering similarity with constraints, there is a tradeoff that should be considered. With constrains, the number of suitable services increases (via 'acceptable matching, as opposed to 'perfect matching'); however, the computation is the cost. In general, this kind of updates is invisible, usually off-line.

(ii)

As previously said, similarity with constrains was introduced to avoid an empty set as an answer to a query. Precision is definitively lower as for 'perfect matching', but the approach allows a flexibility still satisfying a query in a given context and with given constraints.

Finally, precision is less important if the query conditions are satisfied. Additionally, by selecting base on service reputation, the requester has the option to select the precision by adopting a service at a given reputation (we can refer to it as a 'selected precision').

(iii)

Features aggregation are only used for similarity evaluation when not feature matching are found, but an implicit or explicit feature aggregated similarity can be established (see Figure 5.2).

5.6 Conclusion and Future Work

In this chapter, we presented an approach for service invocation using similarity taxonomy with weak, strong, and normal similarity. Practically, services are clustered and service distance/similarity metrics were adopted from text-based domains. A reputation-based mechanism (introduced in [82][93]) is used in combination to context-based similarity and feature relaxation methods to identify a set of services that better serve a given query.

We also introduced the techniques of feature aggregation when similarity is evaluated, and the continuous update of feature classification, i.e., primary/secondary, according to the context. Work can be expanded on these two items, as semantic-based aggregation can be considered.

CHAPTER 6

Proposal validation and case study

Summary

On-line services and social networks became the status nowadays. Collaborating teams on large projects also impose the selection of platform services and appropriate software pieces. With the variety of services and the large spectrum of user's (invoker's) behavior, it is challenging now to select and properly settle services features. In the previous chapters, we proposed, a framework, associated models and algorithms and heuristics to apply them. We partially evaluated the usefulness of our proposals in each chapter and provided appropriate examples, use cases, or local experiments to illustrate the benefits. In this chapter, we revisit more concretely some aspects, illustrate main steps in setting the use of the proposed framework, and provide details on a particular use case on how the proposal is used and show its benefits. Practically, we present a business-oriented platform model, a cost function to be optimized and adjacent methods on how to settle and update service reputation in real cases.

6.1. Forms of on-line services

There are myriads of (almost informal) methods on service selection, delivery, best suitable service, customer satisfaction, customer behavior, and solutions to improve the provider benefit, yet keeping the customer (and user, invoker) satisfied.

We illustrate this need with concrete services and features currently in use, to stress out the relevance of the topic as the support for a more accurate and formal consideration.

We consider three domains, i.e., news, books sales, and on-line software releases to show that service evaluation, hints to customers, and customer loyalty is a major trend in online services. We also comment on particular aspects that are missing, highlighting the need of improvements through new solutions.

6.1.1 News

On news, we identify two main trends on this service category, (i) providing luring news, on how many readers already used the service, or (ii) providing only partial information on a given service (asking implicitly for enrolling, paying, or making an action to get full entry grant). For the first category, we illustrate it via a snapshot from the posting on CNN [96] site, from 11/11/2010.

Log in with Facebook to see your friends' activity | What's this?

- 23,235 people recommended The story behind the Chilean miners' Jesus T-Shirts
- 22,737 people recommended Sony retires the cassette Walkman

- 22,896 people recommended Why students got sick: 'Blackout in a can'
- 24,538 people recommended The $200 million myth
- 21,150 people recommended Rapper talks man down from hotel roof

Figure 6.1. Providing hints to the new (old) customers

The site recommends some articles, without disclosing the readers' profiles, the reason of the recommendation, but only showing the titles of the articles. Also, there are no concerns as to who sees this information; everyone can access and see it. The titles play the role for 'service identification', and have no effect on a reader looking for an article on 'system security', for example. If a ranking is asked at this time, based on the provided content, only those belonging to the contexts associated with the article titles should be considered.

Another way to attract customers is used by social networks or news portals, by not providing the full information on a certain item or service that the user is interested in. Facebook and others, only offer name, pictures, no concrete email, phone number, etc. You must be a subscriber to get complete information. Same approach is taken by the *Portals for science articles*; they only release, *title, abstract, authors*, but no c*onten*t. For getting all the details one either must *subscribe* (with *no fees*), or *pay a subscription fee*, depending on the business model. In the former model, free subscription comes with a model where each subscriber receives *advertizing news* from the service provider (as advertizing is the business support). In the latter, *per item subscription* or *monthly subscription* might be used. The approach might change with the business model.

6.1.2 Books

There are many on-line book selling providers (note, we do not restrict the framework and proposals on on-line service only; we take this examples as the most convenient, without losing the generality).

Focusing on Amazon [94] site, of November 11, 2010, we find, in the context Books, a few statistics offered to the readers, with no details on how the information was obtained. There are categories on *best books of the year/month*, but no criteria, nor reader context (practically, based on the number of books that were sold) are provided. This is quite misleading information for any reader. Also, in another context, there is information on *Books under $50* and *New and used books* that guide the query for a particular book. Such an explicit context avoids a query with constraint on the book price. Extending the service model, the provider offers other book selling features such as *Domestic shipping rates* or *International shipping rates*. Finally, a ranking form the reviewers is provided *[from * to *****]*, but without the reviewers' profile it is hard to digest such indications; this is also driven by the fact that the service provider itself is ranking its own services from the customer side. Finally, additional information on *Learn more*, seen as an interaction feature, might help the user navigate through towards a personalized selection.

6.1.3 Software

Definitively, within the *operational functionalities* (the purpose of the given software), integration interfaces, I/O interfaces, security, accessibility, etc., are basic service features. However, software producers (direct, dealers) offer *Beta version* and *Trial version*, as a first step for building the expected reputation and start a basis of the customers. Based on the most used features in these phases, they do prepare offering *Individual version* (with fewer features, but lower costs) and *Commercial version* (with more features and higher costs). The experiment helps in defining the *primary and secondary features*, in context (here, *personal*, and *corporate*, respectively). Next, the producer builds a *license model* and *upgrade cost model*. This is the basis for ensuring a business model for supporting software sales and maintenance. On maintenance, the *frequency of update* is a feature of the service vector.

Other forms of service ranking are used for on-line content monitoring. For example, ICRA (Internet Content Rating Association) is an independent international organization that rates Internet content with labels ranging from categories including chat, language, nudity, sexual content, violence, gambling, to drugs and alcohol [107]. They are used by major Internet search engines to block or at least advertize the users on the nature of a given content

6.1.4 Target

We try to better understand service selection, based on more formal models, by capturing customer (invoker, either human or agent) behavior and establish ranking with

various targets (benefit, satisfaction, safety, etc.) in various contexts and for different categories of customers.

We also propose an architecture combining service-oriented features and adjacent services along with customer-oriented feedback to settle a business models covering the aspects proposed by the model. As a case study, we target a 'book service', with information gathered from Barnes&Nobles book selling chain, in USA. Some general data were obtained reasonable easier; some were under strict confidentiality for the provider, and only simulations were possible. However, conclusions seem to cover the facets we originally targeted.

6.2. User Participation

Finding services satisfying the user, setting the right costs and delivery conditions is a challenging exercise, as customer behavior/needs are rapidly changing, product lifecycles have a longer adaptability period, including launching window, market penetration, maturity, and development of new features. Having customer (end-user) feedback is crucial for feature setting, costs, delivery, and maintenance. With little differences, on-line service delivery is similar with all other services delivery; however, customer position is easier to express for such kind of services. Delivering personalized services or corporately-endorsed services is usually based on purchase history (visits, purchases, returned goods, campaign responses, etc.).

For an on-line service, some simple user behavior characteristics might settle the 'expected reputation' for a service, e.g., numbers of clicks (visits), duration of stays, frequency of returns, etc. These are used for so-called implicit service rating (and implicitly, service reputation). From a user perspective, implicit reputation is better than explicit reputation, as not biased; if handled on the service provider site, it is less accurate/credible. However, mechanisms for implicit reputation are quiet powerful, such as implicit click-stream data processing, collaborative filtering, and statistical techniques (such as Hidden Markov models).

Explicit user participation is the basis for a more interactive model. Along with it, a support model for customer model (e.g., behavioral patterns), query context, service specification, service reputation, service similarity, primary/secondary service features (from the producer and user point of view), and progressive computation of service reputation are a few of the concepts introduced and developed in the thesis. We presented policies and heuristics on how computations are done for each newly introduced concept. Following, we summarize an architecture that support these concepts, and present some challenging mechanisms when applying it. Finally, we interpret several dependencies between the results obtained when considering a service producer focusing on selling books.

6.3 An Implementation Architecture

In Chapter 4, we presented an on-line/off-line customer-feedback architecture. We expand it now to cover the entire service lifecycle, as a case study.

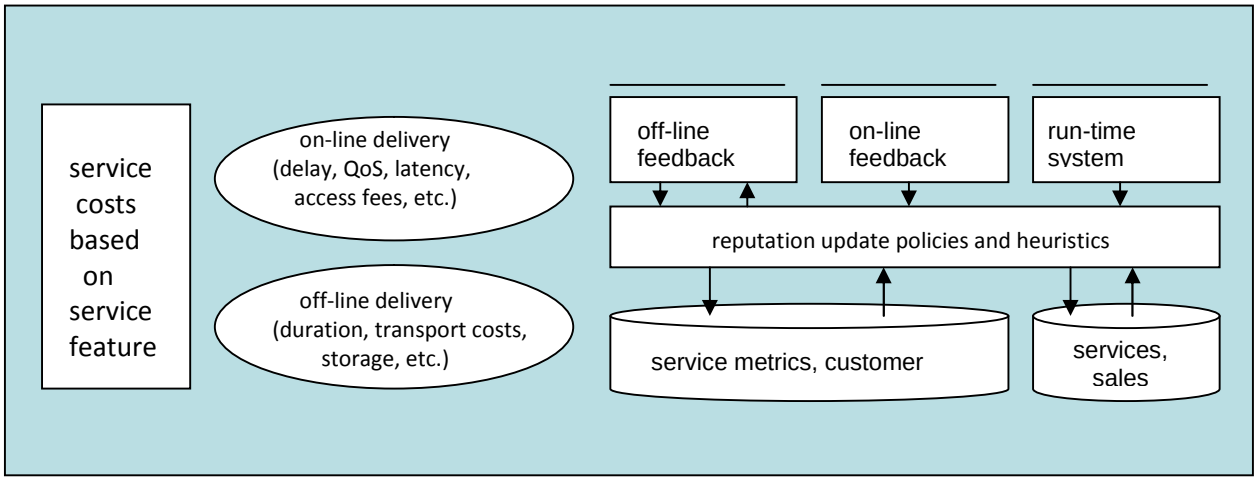


Figure 6.2. Service selection and delivery framework

Figure 6.2 illustrate the three components of the business driving a service model following our approach (service, delivery, customer feedback). The cost function we used for the case study is given by (6.1).

$$\text{COST} = (\#\text{ transactions}) \times \text{Cost}_{\text{service}} + (\#\text{ deliveries}) \times \text{Cost}_{\text{delivery}} + (\#\text{ back-up items}) \text{Cost}_{\text{storage}} + \text{Cost}_{\text{reputation-update}} \quad (6.1)$$

The target is optimizing the COST function (from the producer perspective) considering the customer feedback. While apparently simple, the function hides a lot of dependencies making the optimization very challenging. For example, the *cost related to reputation* is directly related to the number of transactions; also the *cost per service* and *number of transactions* are implicitly related. The number of transactions is related to the

number of deliveries. We introduced expected reputation (from the provider point of view) and feedback reputation, as captured and computed from the users.

For optimizing *the cost per service,* a provider has different actions to take; we considered the following functions:

f1: discount, [temporary]

f2: lower the price [definitively], and

f3: feature update,

while for optimizing the delivery costs, we considered the functions

g1: QoS update

g2: service delivery type updates (urgent, rapid, normal)

The list of functions is open, and decision can be made on other functions as well. We ignored, without losing the generality, the *storage costs*, as the focus was on service reputation and related functions in the cost expressions.

In applying the proposed model, the following aspects are worth to mention as critical for a full implementation: (i) how to establish the initial value for service reputation?, (ii) how to correlate wrongly settled expected reputation?, and (iii) how to establish the primary/secondary features, based on customer feedback. Other mechanisms used in the computation follow the heuristics and the algorithms presented in the previous chapters.

6.3.1 Establishing the 'initial value' for service reputation value

For a given service, we built a service matrix for the service 'books for children', based on customer questionnaire and some non-confidential data for the service provider, as shown in Figure 6.3.

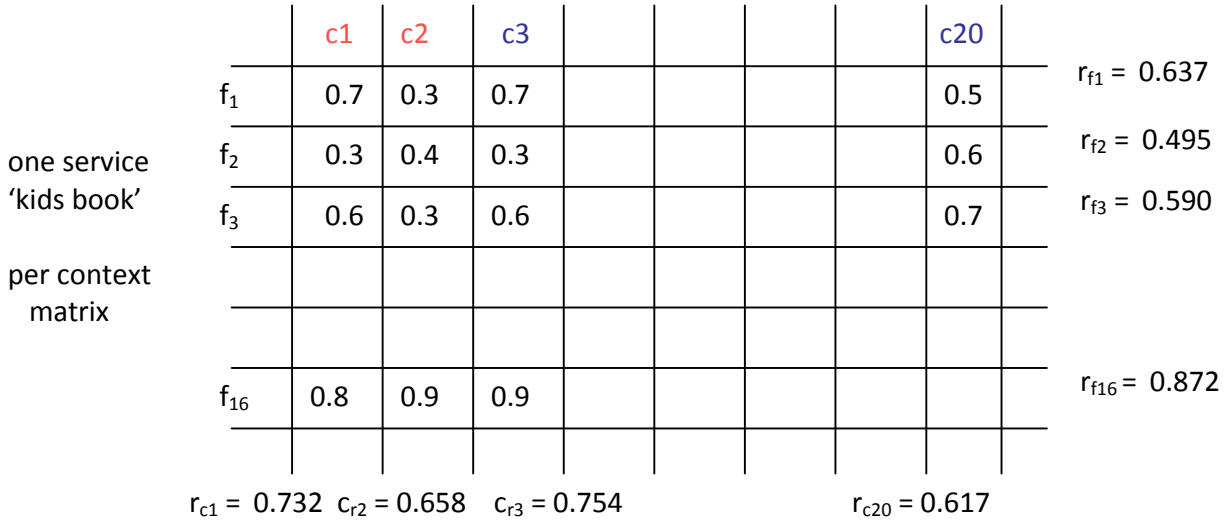


Figure 6.3. Service reputation computation for the service 'kids book'

The matrix shows the ranking per feature and per customer for a new service. The structure of data collected is described in the 'questionnaire legend'. The approach allows computing the service ranking for each feature (vertical column), and for each customer

(horizontal column). By considering the approach presented in Chapter 4, where the customer profile is considered when adapting the service reputation, the reputations service for the above example was determined as $r = 0.703$.

Consequently, when a service is declared as mature, the expected reputation for it is r = 0.7. The updates follow the approach presented in Chapter 4. We highlight a fact that might not be obvious when ranking a service, i.e., miscellaneous functions are sometimes more relevant to the customers. Therefore primary and secondary service should be revisited, as provider view and customer view might be totally disconnected. In this particular case, feature f_3 and f_{16}, apparently not related to the service itself, have a weight in computing service reputation. Another observation was that new customers are not necessarily less important. Definitively, a more refined profile-based customer weight should be considered for a more accurate 'initial estimation' of service reputation.

The presented matrix [M] allowed building two derivate matrices [MP], provider matrix, and [MC], customer matrix for a service. [MP] allows identification of primary and secondary service features based on the provider expectation, while [MC] portrays the primary/secondary features based on customer evaluation. In this particular case, the difference is significant, as originally considered secondary (see, miscellaneous) features in the [MP], two features were ranked as primary in [MC].

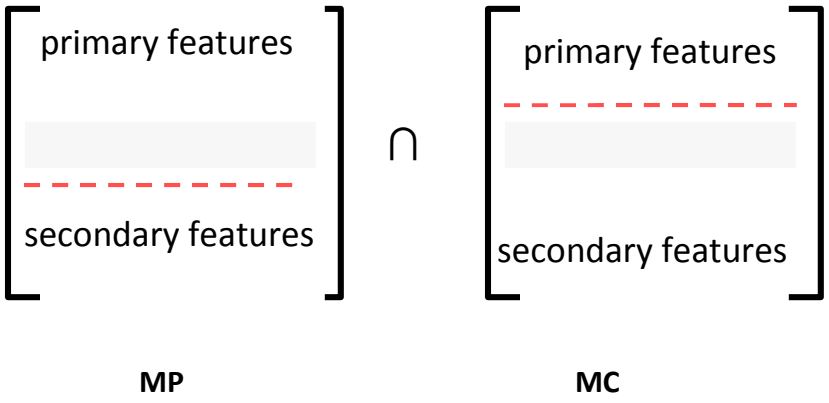


Figure 6.4. Finding secondary/primary dissimilarities between features

We considered MP ∩ MC to identify the discrepancies between MC and MP and focus on those features with a different perception. This had relevance for the design of service features (and implicitly, service costs).

6.3.2 Adapting the reputation

Once a service is offered on the market, the expected reputation should be continuously updated. A business model is usually employed by settling a reasonably increased reputation margin. In the case study, the expected reputation was settled as r = 0.85 (with a 0.15 increased margin) within a reachability windows of T = 45 days (this varies on the type of service). For two book titles (1-3 year old, and 8-10 year old) the variation was significant.

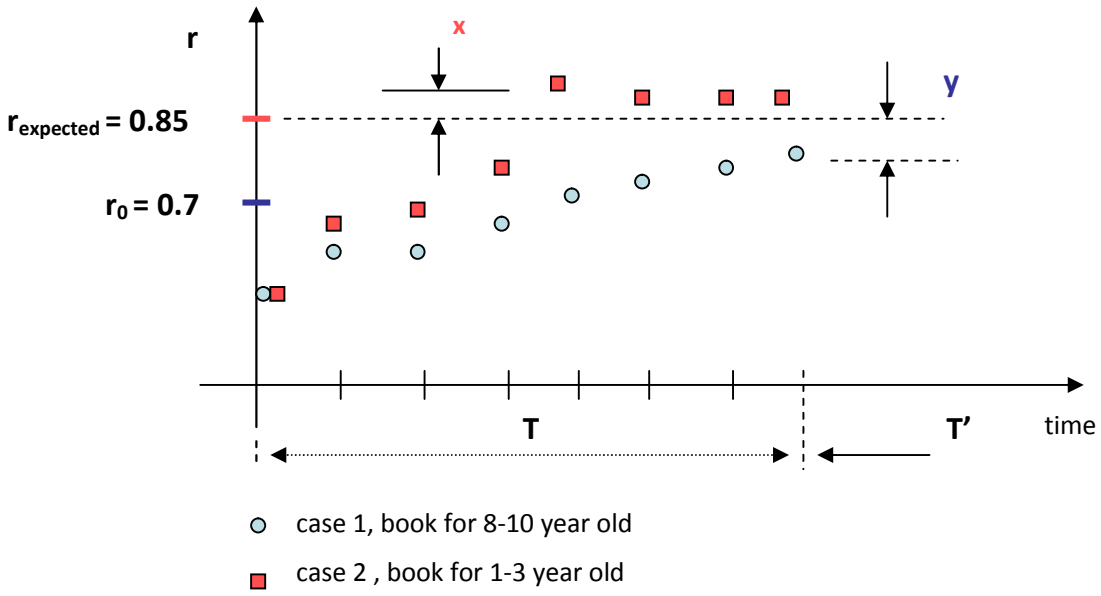


Figure 6.5. Validating the estimated reputation

Based on the data gathered, assuming a uniform serviced reputation across two different contents (1-3 year old, 8-10 year old) (the matrix was built for one context only, containing these two), two variations were observed. The reputation margin was too big for case 1 (by **y**), while for *case 2*, the estimated reputation was exceeded (by **x**). An explanation might be that in pre-school, books are ordered (and bought) by customers (*case 1*), while in *case 2*, the school libraries might fill the need. This is also a hint to the service providers to apply '*discount function*' for some services (*case 1*), or to raise the cost (*case 2*).

6.4. Conclusion

In this Chapter, we presented some case studies on various aspects concerning the applicability and usefulness of the proposed approach, the model and reputation-based services considering new concepts and mechanisms for a better selection of a service.

We reexamined some of the aspects presented in previous chapters, and clarified the main steps for setting the use of the proposed framework. More concretely, a business-oriented platform model was presented together with an optimization of a cost function and afferent methods for settling and updating service reputation in real cases.

CHAPTER 7

Conclusion and Future Work

Achievements

With the amount of information that is available over the Internet, searching for a certain service or data is still difficult. Many available services are similar in terms of what they do, but that is not necessarily enough when it comes to deciding which service to select. Issues such as trust regarding the service and/or the provider need to be taken into consideration when deciding on a service. Also, most of the time the users would like to be able to select a service based on criteria (e.g., specify the importance of a service feature as compared to another feature) and these options are usually limited.

In this thesis, we presented a framework for models and mechanisms for the purpose of validating and enforcing trusted services based on the introduced notions of progressive trust, context-based trust, and context-aware reputation. Along with these mechanisms, the user feedback composition was used for building a dynamically adaptive trust model for Web Services.

We first introduced a framework with appropriate mechanisms that are used for evaluating the services/providers in the light of their respective direct impact on user perception. We continued with an approach for modeling the dynamic user feedback, approach that is to be used for computing the reputation of a service. Next, we proposed a way for selecting services based on distance and similarity. For this purpose, similarity

taxonomy for adjusting service invocation under certain constraints was introduced. To clarify the proposed framework, we presented a few case studies.

Future Work on Open Issues

One aspect that is left out, but worth to be mentioned, is that the reputation adaption function presented in the thesis might be refined. We adopted a linear $r = r_0 (1 + \lambda)$ reputation adaptation function. However, some services might listen to other forms of reputation adaptation function, as $r = r_0 (1 + e^{\lambda/r0})$, or $r = r_0 (1 + \log\lambda/r_0)$, etc. We think that this can be approached by defining on the customer side and on the provider side, a most suitable function for reputation update, considering the type of service and the context.

Another aspect that should be validated with more date sets is related to the duration of the trying period for endorsing the expected reputation. For example, 1-2 moth for a book service, 1-2 weeks for a coffee service, or 6 months for a piece of software. In this case, trying various validation periods might provide a more accurate reputation update.

Service reputation was calculated 'per context'. A global view, from the producer perspective will require studies on weighted cross-context reputation, in the case of a free-context query. For example, a formula might be

$$S_{cross\text{-}context} = (\mathbf{\Sigma}\ w_i \times r_{contexti}) / \mathbf{\Sigma} w_i \qquad (7.1)$$

In this case, large statistics are needed for an optimal tuning of w_i.

Another aspect that was surprising concerned the fact that new customers where quite close to the estimated reputation; this raises the issue of a finer tuning, by considering customer profiles. Heuristics presented din Chapter 2 and Chapter 3 might need updates considering exceptions.

Finally, the fact that some service features that showed strong impact on service reputation were originally classified as secondary (or, even miscellaneous) requires a more customer-oriented service design. Actually, the conclusion is quite the opposite with what is going on with the service launching, when a myriad of features are attached to a service, without a clear evaluation of a need. Even more, new features are added, without accurate validation of the use and customer evaluation of the existing ones.

This is why we consider our contribution as a step ahead for building a solid ‘service science’ targeting both customer comfort and provider benefit.

REFERENCES

1. Y. Joo, “An ODP-Based Type Manager for Trading Services”, http://dpnm.postech.ac.kr/thesis/96/yhjoo/powerpoint.pdf [retrieved August 13, 2010]

2. ANSAware, http://www.ansa.co.uk/ANSATech/ANSAhtml/95-97-websites/brochure.html [retrieved August 13, 2010]

3. CORBA ORB, http://www.cs.wustl.edu/~schmidt/corba-overview.html [retrieved August 13, 2010]

4. ORB Requests, www.omg.org/gettingstarted/corbafaq.htm [retrieved August 14, 2010]

5. M. C. Daconta, L. J. Obrst, and K. T. Smith, “The Semantic Web: A guide to the future of XML, Web Services and Knowledge Management”, June 2003, ISBN 0-471-43257-1

6. O. Dini, “Self-adaptable Trust Mechanism for Web Services”, Master thesis, May 2005, San Jose State University, USA

7. “Web Services architecture overview” available at http://www.ibm.com/developerworks/webservices/library/w-ovr/ [retrieved July 18, 2010]

8. D. Booth, H. Haas, F. McCabe, E. Newcomer, M. Champion, C. Ferris, and D. Orchard, “Web Services Architecture”, 2004, February 11, http://www.w3.org/TR/ws-arch/#technology [retrieved July 18, 2010]

9. T. Bellwood, L. Clement, and C. von Riegen, “UDDI Version 3.0.1”, 2003, October 14, http://www.uddi.org/pubs/uddi_v3.htm [retrieved July 18, 2010]

10. A. Nadalin, C. Kaler, P. Hallam-Baker, and R. Monzillo, “Web Services Security: SOAP Message Security 1.0(WS-Security 2004)”http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf [retrieved July 20th, 2010]

11. S. Anderson, J. Bohren, T. Boubez, M. Chanliau, G. Della-Libera, B. Dixon, et al, “Web Services Trust Language(WS-Trust)”, 2005, February, http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-trust/ws-trust.pdf [retrieved July 20,2010]

12. P. Resnick and H. R. Varian, “Recommender systems”, *Communications of the ACM*, 40(3):56–58, 1997

13. E. Aimeur and F.S. Mani Onana, “Better control on recommender systems”, E-Commerce Technology, 2006. The 8th IEEE International Conference on and Enterprise Computing, E-Commerce, and E-Services, The 3rd IEEE International Conference on Volume, Issue , 26-29 June 2006 pp. 38 – 38

14. T. D. Goldberg, D. Nichols, B. M. Oki, and D. Terry, “Using collaborative filtering to weave an information tapestry”, *Communications of the ACM*, 35(12):61–70, 1992

15. A. Dieberger, P. Dourish, K. Hook, P. Resnick, and A. Wexelblat, “Social navigation: techniques for building more usable system”, *Interactions*, 7, 6, 2000, 36-45.

16. R. Chung, D. Sundaram, and A. Srinivasan, “Integrated Personal Recommender Systems”, ICEC '07 Proceedings of the ninth international conference on Electronic commerce, 2007

17. G. Adomavicius and A. Tuzhilin, “Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions”, IEEE Transactions on Knowledge and Data Engineering, Volume 17, No. 6, June 2005, Pp. 734 – 749

18. P. Massa, and B. Bhattacharjee, “Using Trust in Recommender Systems: An Experimental Analysis”, *Trust Management*, (*Proceedings of the 2nd International Conference*, iTrust 2004).Oxford, UK, LNCS 2995, Springer, pp. 221-235

19. M. Balabanovic and Y. Shoham, “Fab: Content-Based, Collaborative Recommendation”, Comm. ACM, vol. 40, no. 3, pp. 66-72, 1997

20. G. Salton, “Automatic Text Processing”, Addison-Wesley, 1989

21. M. Pazzani and D. Billsus, “Learning and Revising User Profiles: The identification of interesting web sites”, *Machine Learning, 27*, 1997, pp. 313-331

22. G. Linden, B. Smith, and J. York, “Amazon.com recommendations: item-to-item collaborative filtering,” *IEEE Internet Computing*, vol. 7, no. 1, 2003, pp. 76–80

23. X. Su and T.M. Khoshgoftaar, “A survey of collaborative filtering techniques”, Advances in Artificial Intelligence, p.2-2, January 2009

24. G. Shani, D. Heckerman, and R. I. Brafman, “An MDP-based recommender system,” *Journal of Machine Learning Research*, vol. 6, pp. 1265–1295, 2005.

25. T. Hofmann, “Latent semantic models for collaborative filtering,” *ACM Transactions on Information Systems*, vol. 22, no. 1, pp. 89–115, 2004

26. A. Popescu, L. Ungar, D.M. Pennock, and S. Lawrence, “Probabilistic models for unified collaborative and content-based recommendation in sparse-data environments”, In *Proceedings of the Seventeenth Conference on Uncertainity in Artificial Intelligence*, 2001.

27. M. Pazzani, “A Framework for Collaborative, Content-Based, and Demographic Filtering”, Artificial Intelligence Rev., pp. 393-408, Dec. 1999.

28. A.I. Schein, A. Popescu, L.H. Ungar, and D.M. Pennock, “Methods and Metrics for Cold-Start Recommendations”, Proc. 25th Ann. Int’l ACM SIGIR Conf., 2002

29. R. Burke, “Hybrid recommender systems: survey and experiments,” *User Modelling and User-Adapted Interaction*, vol. 12, no. 4, pp. 331–370, 2002

30. M. Balabanović and Y. Shoham, “Content-based, collaborative recommendation”, *Communications of the ACM*, vol. 40, no. 3, pp. 66–72, 1997

31. B. M. Sarwar, J. A. Konstan, A. Borchers, J. Herlocker, B. Miller, and J. Riedl, “Using filtering agents to improve prediction quality in the grouplens research collaborative filtering system”, in *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW '98)*, Seattle, Wash, USA, 1998, pp. 345–354

32. P. Melville, R.J. Mooney, and R. Nagarajan, “Content-Boosted Collaborative Filtering for Improved Recommendations”, Proc. 18th Nat’l Conf. Artificial Intelligence, 2002

33. M. Claypool, A. Gokhale, T. Miranda, et al., “Combining content-based and collaborative filters in an online newspaper”, in *Proceedings of the SIGIR Workshop on Recommender Systems: Algorithms and Evaluation*, Berkeley, Calif, USA, 1999

34. J. A. Delgado, “Agent-based information filtering and recommender systems on the internet”, Ph.D. thesis, Nagoya Institute of Technology, February 2000

35. X. Su, R. Greiner, T. M. Khoshgoftaar, and X. Zhu, “Hybrid collaborative filtering algorithms using a mixture of experts”, in *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence (WI '07)*, Silicon Valley, California, USA, November 2007, pp. 645–649

36. D. Billsus and M. Pazzani, “User Modeling for Adaptive News Access,” User Modeling and User-Adapted Interaction, vol. 10, nos. 2-3, pp. 147-180, 2000

37. T. Tran and R. Cohen, “Hybrid Recommender Systems for Electronic Commerce,” Proc. Knowledge-Based Electronic Markets, Papers from the AAAI Workshop, Technical Report WS-00-04, AAAI Press, 2000

38. T. Grandison, “Trust Management for Internet Applications”, PhD thesis, Imperial College London, 2003

39. A. Jøsang, R. Ismail, and C. Boyd, “A survey of trust and reputation systems for online service provision”, Decision Support Systems, v.43 n.2, p.618-644, March, 2007 [doi>10.1016/j.dss.2005.05.019]

40. A. Abdul-Rahman and S. Hailes, “Supporting trust in virtual communities”, In HICSS 2000, vol. 6, 2000, pp.6007

41. A. Bonatti, N. Shahmehri, C. Duma, D. Olmedilla, W. Nejdl, M. Baldoni, C. Baroglio, A. Martelli, V. Patti, P. Coraggio, G. Antoniou, J. Peer, and N. E. Fuchs, “Rule-based policy specification: State of the art and future work”, Technical report, Working Group I2, EU NoE REWERSE, August 2004, http://rewerse.net/deliverables/i2-d1.pdf [retrieved July 30, 2010]

42. O. Hasan, L. Brunie, J.-M. Pierson, and E. Bertino, “Elimination of Subjectivity from Trust Recommendation”, Trust Management III (IFIP Advances in Information and Communication Technology, 2009),Volume 300/2009, 65-80

43. B. Christianson and W. S. Harbison, “Why isn’t trust transitive?”, Security Protocols International Workshop, 1996

44. G. Caronni, “Walking the web of trust”, In WETICE, 2000

45. N. Li, B. N. Grosof, and J. Feigenbaum, “Delegation logic: A logic-based approach to distributed authorization”, ACM Trans. Inf. Syst. Secur., 6(1):128–171, 2003

46. A. Jøsang and S. Pope, “Semantic constraints for trust transitivity”, Proceedings of the 2nd Asia-Pacific conference on Conceptual modeling, January 01, 2005, Newcastle, New South Wales, Australia, p.59-68

47. C.M. Jonker, J.J.P. Schalken, J. Theeuwes, and J. Treur, “Human experiments in trust dynamics”, *Trust Management*, (*Proceedings of the 2nd International Conference*, iTrust 2004), Oxford, UK, LNCS 2995, Springer, pp 206-219

48. S. Staab, B.K. Bhargava, L. Lilien, A. Rosenthal, M. Winslett, M. Sloman, T. S. Dillon, E. Chang, F.K. Hussain, W. Nejdl, D. Olmedilla, and V. Kashyap, “The pudding of trust”, *IEEE Intelligent Systems*, 19(5):74-88, 2004

49. M. Blaze, J. Feigenbaum, and J. Lacy. “Decentralized Trust Management”, in *IEEE Conference on Security and Privacy*,Oakland, California, USA, 1996, http://www.crypto.com/papers/policymaker.pdf, [retrieved August 15, 2010]

50. A. Jøsang and N. Tran, “Trust Management for E-Commerce”, 2000, http://citeseer.nj.nec.com/375908.html, [retrieved August 15, 2010]

51. K. Seamons , M. Winslett , T. Yu , B. Smith , E. Child , J. Jacobson , H. Mills , and L. Yu, “Requirements for Policy Languages for Trust Negotiation”, Proceedings of the 3rd International Workshop on Policies for Distributed Systems and Networks (POLICY'02), June 05-07, 2002, pp 68

52. W. Negm, “Bringing Balance to Web Services Solving the Trust & Threat Equation”, *Network Computing*, January 2004, http://www.forumsys.com/resources/resources/whitepapers/04_Bringing_Balances_Security.pdf, [retrieved August 18, 2010]

53. C. Kaufman, R. Perlman, and M. Speciner, “Private Communication in a Public World”. *Network Security,* New Jersey: Prentice Hall PTR, 2002

54. N. Li , J.C. Mitchell , and W.H. Winsborough, “Design of a Role-Based Trust-Management Framework”, Proceedings of the 2002 IEEE Symposium on Security and Privacy, May 12-15, 2002, p.114

55. W. Nejdl, D. Olmedilla, and M. Winslett, “PeerTrust: automated trust negotiation for peers on the semantic web”, In Workshop on Secure Data Management in a Connected World (SDM’04), Toronto, Aug. 2004

56. P. Bonatti and P. Samarati, “Regulating Service Access and Information Release on the Web,” *7th ACM Conference on Computer and Communications Security*, Athens, Greece, November 2000

57. A. Herzberg, J. Mihaeli, Y. Mass, D. Naor, and Y. Ravid, “Access Control Meets Public Key Infrastructure, Or: Assigning Roles to Strangers,” *IEEE Symposium on Security and Privacy*, Oakland, CA, May 2000

58. M. Czenko, S. Etalle, D. Li and W.H. Winsborough “An Introduction to the Role Based Trust Management Framework RT”, Foundations of Security Analysis and Design IV(LNCS), 2007, Volume 4677/2007, 246-281

59. C. Dellarocas and P. Resnick, “Online Reputation Mechanisms: A Roadmap for Future Research”, Summary report of the First Interdisciplinary Symposium on Online Reputation Mechanisms, April 26-27, 2003

60. C. Dellarocas, “Reputation Mechanisms”, Handbook on Information Systems and Economics, T. Hendershott (ed.), Elsevier Publishing, 2006, 629-660

61. K. Aberer and Z. Despotovic, “Managing Trust in a Peer-2-Peer Information System”, *In Proc. of the IX International Conference on Information and Knowledge Management,* Atlanta, Georgia, 2001

62. S. Kamvar and M. Schlosser, “The EigenTrust Algorithm for Reputation Management in P2P Networks”, *WWW, Budapest, Hungary*, 2003

63. L. Xiong and L. Liu, "PeerTrust: Supporting Reputation-Based Trust for Peer-to-Peer Electronic Communities," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 7, July 2004, pp. 843-857

64. J. Liu and V. Issarny, “Enhanced Reputation Mechanism for Mobile Ad Hoc Networks”, *Trust Management*, (*Proceedings of the 2nd International Conference*, iTrust 2004), Oxford, UK, LNCS 2995, Springer, pp. 48-62

65. J. Sabater and C. Sierra, “REGRET: A reputation Model for Gregarious Societies”, *4th Workshop on Deception*, *Fraud and Trust in Agent Societies*, 2001

66. L. Mui, M. Mohtashemi, C. Ang, P. Szolovits, and A. Halberstadt, “Ratings in Distributed Systems: A Bayesian Approach”, *11th Workshop on Information Technologies and Systems (WITS)*, New Orleans, 2001

67. C. Bertocco, “Context-dependent Reputation Management in Multi-Agent Systems”, Doctoral thesis, 2009. http://paduaresearch.cab.unipd.it/1695/1/bertocco-cristian-thesis.pdf, [retrieved on August 27, 2010]

68. L. Mui, “Computational Models of Trust and Reputation: Agents, Evolutionary Games, and Social Networks”, Ph.D. Dissertation, Massachusetts Institute of Technology, 2003, http://groups.csail.mit.edu/medg/ftp/lmui/computational%20models%20of%20trust%20and%20reputation.pdf. [retrieved on August 30, 2010]

69. Y. Wang and J. Vassileva, “Toward trust and reputation based Web service selection: A survey”, Int. Trans. Syst. Sci. Appl. 3, 2, 2007, 118--132

70. eBay, accessible at http://www.ebay.com

71. B. Dragovic, E. Kotsovinos, et al., “XenoTrust: Event-based distributed trust management”, Second International Workshop on Trust and Privacy in Digital Business, Prague, Czech Republic, 2003

72. P.-Y. Chen, Y.-C. Chou, R.J. Kauffman, “Community-Based Recommender Systems: Analyzing Business Models from a Systems Operator's Perspective”, Proceedings of the 42nd Hawaii International Conference on System Sciences – 2009, pp. 1-10

73. T.P. Novak and D.L. Hoffman, “A conceptual framework for considering web-based business models and potential revenue streams”, *International Journal of Marketing Education*, 1, 1, 2004, 33-43

74. Y.B. Chen and J.H. Xie, “Online consumer reviews: a new element of marketing communications mix”, Working paper, Eller College of Management, University of Arizona, Tucson, AZ, 2004

75. N. Kumar and I. Benbasat, “The influence of recommendations and consumer reviews on evaluations of Web sites”, Information Systems Research, 17, 4, 2006, 425-439

76. O. Dini, M. Moh, and A. Clemm, “Web Services: Self-adaptable Trust Mechanisms”, AICT/SAPIR/ELETE 2005, pp.83-89

77. R. Jurca and B. Faltings. “An Incentive Compatible Reputation Mechanism”, Proc. IEEE Conf. on E-Commerce, Newport Beach, CA, June 2003, pp. 285-292

78. K.-J. Lin, H. Lu, T. Yu, and C.-e. Tai, “A reputation and trust management broker framework for Web applications”, Proceedings of The 2005 IEEE International Conference on e-Technology, e-Commerce and e-Service. April 2005. pp. 262-269

79. C. Dellarocas, “Strategic manipulation of Internet opinion forums: implications for consumers and firms”, Mgmt. Sci., 52, 10, 2006, 1577-1593

80. P. Varga, G. Kún, G. Sey, I. Moldován, and P. Gelencsér, "Correlating User Perception and Measurable Network Properties: Experimenting with QoE", in Autonomic Principles of IP Operations and Management, 2006, pp. 218-221

81. E. Ibarrola, F. Liberal, I. Taboada, R. Ortega, "Web QoE Evaluation in Multi-agent Networks: Validation of ITU-T G.1030," ICAS, 2009 Fifth International Conference on Autonomic and Autonomous Systems, 2009, pp.289-294

82. O. Dini, P. Lorenz, and H. Guyennet, “An Enhanced Architecture for Web Recommenders”, SERVICE COMPUTATION 2009, IEEE Press, Athens, Greece, pp. 372 – 378, ISBN: 978-1-4244-5166-1

83. “Mean opinion score and metrics”, http://technet.microsoft.com/en-us/library/bb894481.aspx [retrieved Jan 10, 2010]

84. PESQ, PESQ Algorithm firmware, http://www.qoesystems.com/products/pesq_basic.htm [retrieved January 13, 2010]

85. R.M. Bruckner, B. List, and J. Schiefer, “Striving Towards Near Real-Time Data Integration for Data Warehouses”, In Proc. of the 4th Intl. Conf. on Data Warehousing and Knowledge Discovery (DaWaK 2002), Springer LNCS 2454, Aix-en-Provence, France, Sept. 2002, pp. 317–326

86. J. Schiefer, A. Seufert, “Management and Controlling of Time-Sensitive Business Processes with Sense & Respond”, In Proceedings of International Conference on Computational Intelligence for Modelling Control and Automation, Vienna, Austria, 2005

87. S. Cost and S. Salzberg, “A Weighted Nearest Neighbor Algorithm for Learning Symbolic Features”, Machine Learning, No. 10, 1993, pp. 57-78

88. L.S. Larkey and W. Croft, “Combining Classifiers in text Classifications Techniques”, ACM SIGIR 1998

89. H.-H. Do and E. Rahm, “COMA – A System for flexible Combination of Schema Matching Approaches”, VLDB 2002

90. A.M. Zaremski and J.M. Wing, “Specification matching of software components”, TOSEM, No. 6, 1997, pp. 333-369

91. X. Dong, A. Halevy, J. Madhavan, E. Nemes, and J. Zhang, “Similarity Search for Web Services”, The 30th VLDB Conference, Toronto, 2004

92. C. Bouras and V. Tsogkas, “Improving text summarization using noun retrieval techniques”, LNCS, Knowledge-based Intelligent Information and Engineering Systems, vol. 5178/2008, pp. 593-600

93. O. Dini, P. Lorenz, A. Abouaissa, and H. Guyennet, “Dynamic Feedback for Service Reputation Updates”, ICAS 2010, Cancun, Mexico, pp. 168-175, ISBN: 978-1-4244-5915-5

94. Amazon, Books, http://www.amazon.com/ [retrieved on November 11, 2010]

95. O. Dini, P. Lorenz, A. Abouaissa, and H. Guyennet, “Online Service Similarities and Reputation-based Selection”, SERVICE COMPUTATION 2010, Lisbon 2010

96. CNN News http://www.cnn.com/ [retrieved on November 11, 2010]

97. P. Yolum and P. M. Singh, “Emergent Personalized Communities in Referral Networks”, IJCAI Workshop on Intelligent Techniques for Web Personalization (ITWP), 2003

98. L.-H. Vu, M. Hauswirth, and K. Aberer, “Towards P2P-based Semantic Web Service Discovery with QoS Support”, Proceeding of Workshop on Business Processes and Services (BPS), Nancy, France, 2005

99. E. Damiani, C. Vimercati, S. Paraboschi, P. Samarati, and F. Violante, “A Reputation-Based Approach for Choosing Reliable Resources in Peer-to-Peer Networks”, In Proceedings of the 9th ACM conference on Computer and Communications Security (CCS’02), ACM, 2002 pp 207–216

100. P. Van Eijk and M. Diaz, “Formal Description Technique Lotos: Results of the Esprit Sedos Project”, Elsevier Science Inc., New York, NY, 1989

101. J. M. Spivey, The Z notation: a reference manual, Prentice-Hall, Inc., Upper Saddle River, NJ, 1989

102. S.-H. Chae, J.-W. Baek, M.-S. Jeong, J.-T. Park, S.-B. Kim, and C.-K. Hwang, "Implementation of GDMO to IDL Translator and CORBA/CMIP Gateway for TMN/CORBA Integration", APNOMS'98, Sendai, Japan, September 1998, pp.119-130

103. H. Zheng and S.-x. Li, “The Description of CORBA Objects Based on Petri Nets”, Formal Methods and Software Engineering (LNCS 2002), Volume 2495/2002, 48-56

104. T. Bray, J. Paoli, C.M. Sperberg-McQueen, E. Maler, and F. Yergeau, (November 2008), Extensible Markup Language (XML) 1.0 (Fifth Edition). World Wide Web Consortium (W3C), http://www.w3.org/TR/REC-xml/ [retrieved August 5, 2010]

105. E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana, “Web Services Description Language (WSDL) 1.1”, March 2001, http://www.w3.org/TR/wsdl, [retrieved August 5, 2010]

106. M. Gudgin, M. Hadley, N. Mendelsohn, J.-J. Moreau, H.F. Nielsen, A. Karmarkar, and Y. Lafon (2007, 27 April), SOAP Version 1.2 Part 1: Messaging Framework (Second Edition), http://www.w3.org/TR/soap12-part1/ [retrieved August 10, 2010]

107. ICRA, http://www.fosi.org/icra/, [retrieved November 1, 2010]

My publications

International journal

1. O. Dini, P. Lorenz, A. Abouaissa, H. Guyennet, "A Framework for Progressively Trusting Services", International Journal on Advances in Intelligent Systems, Vol. 3, no. 3&4, 2010, ISSN: 1942-2652

International Conference

2. O. Dini, P. Lorenz, A. Abouaissa, H. Guyennet, "Online Service Similarities and Reputation-based Selection", Second International Conferences on Advanced Service Computing, SERVICE COMPUTATION'10, November 21-26, 2010, Lisbon, Portugal

3. O. Dini, P. Lorenz, A. Abouaissa, H. Guyennet, "Dynamic Feedback for Service Reputation Updates", Sixth International Conference on Autonomic and Autonomous Systems, ICAS'10, March 7-13, 2010, Cancun, Mexico, pp. 168-175.

4. O. Dini, P. Lorenz, H. Guyennet, "An Enhanced Framework for Web Recommenders", First International Conferences on Advanced Service Computing, SERVICECOMP'09, November 15-20, 2009, Athens, Greece, pp. 372-378.

.

.