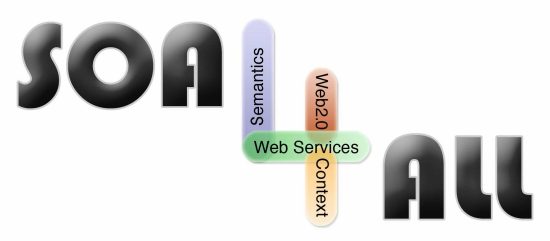

Project Number:	**215219**
Project Acronym:	**SOA4All**
Project Title:	**Service Oriented Architectures for All**
Instrument:	**Integrated Project**
Thematic Priority:	**Information and Communication Technologies**

D4.1.1 Contextual Service Adaptation Framework

Activity N:	2 – Core R&D Activities
Work Package:	4 – Service in Context
Due Date:	M6
Submission Date:	15/09/2008
Start Date of Project:	01/03/2006
Duration of Project:	36 Months
Organisation Responsible of Deliverable:	Open University
Revision:	1.0
Author(s):	Carlos Pedrinaci (OU), Nikolay Mehandjiev (UNIMAN), Carlos Ruiz Moreno (iSOCO), Pierre Grenon (OU)
Reviewer(s):	Marc Richardson (BT), Tomas Vitvar (UIBK)

Project co-funded by the European Commission within the Seventh Framework Programme (2007-2013)		
Dissemination Level		
PU	Public	**X**
PP	Restricted to other programme participants (including the Commission)	
RE	Restricted to a group specified by the consortium (including the Commission)	
CO	Confidential, only for members of the consortium (including the Commission)	

Version History

Version	Date	Comments, Changes, Status	Authors, contributors, reviewers
0.1	21/05/2008	First skeleton of the document	Carlos Pedrinaci (OU)
0.2	16/06/2008	Improve vision	Carlos Pedrinaci (OU)
0.3	01/07/2008	Include overall framework based on Heuristic Classification and Parametric design. State of art	Carlos Pedrinaci (OU)
0.4	08/07/2008	Merged contributions from iSOCO and UNIMAN	Nikolay Mehandjiev (UNIMAN), Carlos Ruiz Moreno (iSOCO), Carlos Pedrinaci (OU)
0.5	15/07/2008	Contribution from iSOCO	Carlos Ruiz Moreno (iSOCO)
0.6	22/07/2008	Contribution from UNIMAN	Nikolay Mehandjiev (UNIMAN)
0.7	29/07/2008	Merged contributions, Edited current version	Carlos Pedrinaci (OU)
1.0	06/08/2008	Final edits	Carlos Pedrinaci (OU)
1.1	22/08/2008	Addressed Marc Richardson's comments	Carlos Pedrinaci (OU)
1.2	11/09/2008	Addressed Tomas Vitvar's comments	Carlos Pedrinaci (OU), Pierre Grenon (OU), Nikolay Mehandjiev (UNIMAN)

Table of Contents

List of Figures

Glossary of Acronyms

Acronym	Definition
D	Deliverable
EC	European Commission
GUIs	Graphical User Interface
ISP	Internet Service Provider
KOPE	Knowledge-Oriented Provenance Environment
OWL	Ontology Web Language
PSM	Problem-Solving Methods
QoS	Quality of Service
RDF(S)	Resource Description Framework Schema
SAWSDL	Semantic Annotations for Web Service Description Language
SME	Small and Medium-sized Enterprises
SOA	Service-Oriented Architecture
TMDA	Task Method Domain Application
WP	Work Package
WSDL	Web Service Description Language
WSMO	Web Service Modelling Ontology

Executive summary

This deliverable documents the overall context adaptation framework, which will guide and integrate work on the different tasks within WP4: Context Adaptation of SOA4All. The framework will no doubt evolve with the detailed design of the different aspects within individual tasks, so this document should be seen as an initial specification providing informing foundation and a roadmap, rather than as a final summary of the results from this WP. This is consistent with its timing as a part of Milestone 1 of the SOA4All project.

SOA4All aims to open up service construction, discovery and consumption to the general public, and provide the mechanisms that can support a web of semantically enriched services at a massive and global scale. The latter part of the aim implies that services should be adaptable to the specific context of their consumption and use. The former part of the aim suggests that this context adaptation should be automated, together with the gathering and processing of context-related information.

Existing work in context adaptation is usually focused on readily available context information such as location, time or level of environmental noise, creating specialised context adaptation mechanisms, mainly within the domain of mobile applications. In contrast to them, SOA4All's envisioned approach to context adaptation aims to consider a wide variety of context factors at different levels of granularity, and to provide generic adaptation facilities using dynamic context recognition, and service parameterisation features.

The following theoretical and architectural underpinnings from the general area of Knowledge Based Systems are chosen to enable automatic context adaptation:

- Heuristic Classification is the overall conceptual approach that will be used to process context-related values from the environment into a consistent and sufficiently abstract set of context parameters that can be used to heuristically determine the relevant context at hand. The context recognised is used input for the subsequent adaptation of services;
- Parametric Design principles are used to construct services in a way allowing the incorporation of context-specific variations of their behaviour;
- A distributed context repository will provide the means to seamlessly store and retrieve distributed, heterogeneous, dynamic and sophisticated context information.
- Context information will be expressed in terms of a Context Ontology stack, which will be aligned with WSMO, the service modelling ontology used in SOA4All, and with the task specific, yet domain independent, heuristic classification and parametric design ontologies. The ontology stack will be driven by an established set of general requirements such as completeness, scalability and granularity, and validated by the applicability to the various use cases of the project.
- Web 2.0 principles will support the use of community-provided and derived context information. Information gathered concerning users and services behaviour will be processed in order to profile both users and services.

The Contextual Services Adaptation Framework outlined herein integrates these theoretical underpinnings with the specific requirements gleaned from the initial draft versions of the SOA4All case studies, and establishes the ways in which the WP4 results will be evaluated. Additionally, this deliverable appraises the current state-of-art in context adaptation to outline the progress envisioned over the state-of-art, and outlines how context adaptation fits with the overall architecture underpinning SOA4All. Finally, the deliverable also establishes further details about the mechanisms underlying the framework where appropriate, in order to provide a sufficiently detailed roadmap for further research and development work within the work package.

1. Introduction

SOA4All intends to enable a world where billions of users are exposing and consuming services, providing an advanced platform that integrates four pillars: Web principles, Web 2.0, Semantic Web and Context. In general, context management is a widely important topic in Information Systems, traditionally concerned with the impact that contextual information has, or rather should have, on the behaviour of the system because of agents (humans or software) interaction and preferences, and additional information characterising the situation at hand.

In SOA4All, the impact of context management is crucial as a way to facilitate the customization of existing services for the needs and expectations of users. Thus, a key for success is to define services in such a way that they can be adapted to specific user context making services more usable by broadening their applicability. Moreover, nowadays taking into account contextual factors is increasingly seen as a prerequisite for appropriate service use and successful dynamic composition. These contextual factors range from immediate concerns of location and language to legal issues and financial regulations.

This deliverable outlines the Contextual Service Adaptation Framework, supports the management of contextual information, the recognition of relevant contextual knowledge about services and users, and eventually the adaptation of existing services during the composition and execution phases according to recognised contexts.

The remainder of the document is organized as follows: Section 3 surveys related work in Context Adaptation for Services, Context Modelling and Frameworks. Section 3 describes our overall vision on Contextual Service Adaptation. Section 4 proposes in detail our Contextual Service Adaptation Framework. Finally, Section 5 concludes the deliverable.

1.1 Methodology

This document describes the result of the first task of WP4: the development of a conceptual integration framework, which will serve as a blueprint integrating the work on the remaining tasks of the work package. The contribution of the work carried out within the first task to the overall framework is to have:

(a) Established the definition and scope of context to be considered by SOA4All.

(b) Established the shortcomings in existing context adaptation frameworks, which prevent direct reuse of their results on SOA4All.

(c) Selected theoretical approaches to underpin context derivation, context recognition and service adaptation.

(d) Established the overall context adaptation vision, supported by cross-references to and from the use cases developed under WP7, 8 and 9.

(e) Developed the criteria and initial design decisions on the context management service and an overall view on context and how it could be represented semantically.

(f) Clarified the social dimensions of specifying and managing context and followed through the resulting requirements to creating a modular and distributed context repository.

(g) Identified the possible suitability of Semantic Spaces as the approach to providing such a context repository.

(h) Developed in some detail the application of heuristic classification to context recognition and the application of parametric design to context adaptation.

The results of this first stage will now feed into systematic processes of developing an ontological treatment of context, a context management service, a context recognition service and a context adaptation mechanism. Most tasks in this process are preceded by a common task of creating extensions of WSMO Goals thus integrating context adaptation with the semantic service stack. This is followed by two iterations of each task, ensuring propagation of results from the first iteration of each task to the second iteration of the other relevant tasks.

1.2 Alignment with the architecture of the project

Currently the architecture of the project is in a very early stage and it is therefore hard to describe herein a complete alignment of the Contextual Service Adaptation Framework with the overall architecture. There are however some aspects that can be identified regarding the integration of the framework with the rest of the components of the project:

- Semantic Spaces has been identified as a possible and quite plausible infrastructure for capturing contextual heterogeneous, dynamic and distributed information over the Web.
- A Context Recognition service will be developed and provided to the rest of the architecture in order to support identifying relevant contextual information in order to adapt the behaviour of applications and services.
- The Provisioning platform will be making use of the infrastructural services provided by the Contextual Service Adaptation Framework in order to adapt the UI, as well as to support the use of contextual information while defining composite services. Conversely, the Provisioning platform will provide valuable contextual information about users and services, thus creating a virtuous circle contributing to the gradual enhancement of the services provided.
- The Consumption platform will also make use of the infrastructural services. In this case, emphasis will be put on the use of the functionality offered by the framework at runtime. Additionally the Consumption platform will provide valuable contextual information about users and services.
- The Monitoring infrastructure will be generating a large amount of monitoring information that will represent a highly valuable source of information for the Contextual Service Adaptation Framework.

1.3 Alignment with the use cases

In this section we cover the different use cases contemplated within SOA4All and identify the main aspects where the Contextual Service Adaptation Framework will play an important role. Like in the previous case, the understandable immaturity of use cases at this stage prevents us from performing a more detailed analysis.

1.3.1 End-user Integrated Enterprise Service Delivery Platform

This use case is based on a set of complementary scenarios from the public sector, centered on the new EC Services Directive. The core scenario is about the administrative processes involved in opening a new Munich branch of a Madrid-registered business. This is extended by scenarios of a “single point of contact”, “alcohol license” and “registering residency status”.

One of its deliverables is a novel tool with an intuitive user interface for lightweight process and service composition and consumption, the other one is a “virtualization” layer in order to close the usability gap between ordinary end users and state-of-the-art enterprise process

development and service delivery

The needs for context adaptation in relation to this case study are as follows:

- Services presented to citizens and civil servants are to be filtered according to user profile. The relevant context factors will include purely personal factors such as age and sex, language and computer skills, but also some organizational factors such as the role of the civil servant, and relevant rules and regulations.
- The services presented to citizens and civil servants will also be filtered according to the task undertaken and prioritized according to previous choices and traces of interaction between a citizen and the system.
- Services will be adapted to suit the context of service consumption/use. For example, the service of opening a new business branch in Munich will take into account local legislation in the state of Bavaria, town regulations in Munich and procedures particular to the local chamber of commerce.
- A user should be notified about the nature of the context adaptation which can be applied to a service or a composite service. Then he or she should be able to select a (sub) set of adaptations that may occur at runtime.

The generation and storage of the relevant context information, and even the specification of what context factors are relevant to the target domain in question will happen in a fully distributed manner. For example the context regarding history of previous service use will be generated and tracked by the monitoring infrastructure, whilst the contextual information about regulations and procedures applicable to the Munich locality will be provided by Munich-based service providers such as accountants, or by civil servants or lawyers employed by the Munich Town Hall.

1.3.2 W21C BT Infrastructure

The BT W21C case study aims to provide user-friendly facilities for advanced service discovery, composition, consumption and monitoring using the existing Web21c infrastructure exposing core BT communication capabilities. The first scenario is about designing a simple application by composing existing services, and the second scenario is about enabling Bulk Resellers to implement innovative business ideas using unbranded services provided by BT, or to integrate said services into their business processes.

The following context adaptation features are foreseen in this case study:

- Information about the current location of the user will be used to select the best-priced services, and to decide on, and implement, the cheapest routing of calls.
- The type of task and user role (i.e. business or pleasure) will be used to select appropriate services and setting quality of service parameters (QoS) of a call. Indeed, business users will require higher QoS when they make a business call than for their personal calls.
- Information about time is routinely used to determine and set different tariffs (e.g. peak/off peak/weekend tariffs).
- Information about the social environment and presence of other users will be used to determine parameters of the call and the best routing of a call. For example, if the call receiver is in an important meeting, the system will adapt the delivery service to either convert the message to SMS or route it to voicemail.
- Context adaptation mechanisms are also envisaged for configuring payment services to the accepted modes in a given country and to the applicable VAT rate for that country, whilst the video and audio streaming services are adapted to the type of

target device and the bandwidth available.

The generation of context information is distributed between user's devices and the communication network equipment, with higher reliance on automatically generated data and comparatively lower reliance on high-granularity context information specified by human input. For example, automatically stored data about service executions will enable automatic derivation of data about the QoS. In terms of higher-granularity context, information about communication patterns within social networks would also be a valuable context factor. The context adaptation functionality will also be distributed between user devices, performing run-time adaptation to levels of noise, for example, and the communication network, which can perform operations such as context-dependent routing of the call.

1.3.3 C2C Service eCommerce

This case study integrates three complementary scenarios: service-based marketplace for ISPs, an innovative web shop based on a digital video channel linking all actors through the supply chain (producers, distributors, customers), and a service-based platform for creating TV programming by a community of users. They will be eventually integrated by using the service-based marketplace as a common platform, and providing community-generated video content through the web shop.

The following context adaptation features are currently foreseen in relation to this case study:

- The interfaces and available tools will be adapted to the personal profiles of the current user, including language and skill levels, after classifying him or her in a number of categories.
- The video distribution channel will be adapted to the target device and communication bandwidth.
- The financial services on the service-based marketplace can be sorted according to the accepted mechanism of payment in the locality of the user (for example credit cards in the UK, postal order in Eastern Europe, bank account information in Germany), and the appropriate VAT rate for the given type of service according to the current legislation and European and country levels.
- The financial procedures will also be adapted to the profile of users and providers, for example mandatory credit rating checks can be introduced as appropriate.
- Context-specific information about location and the applicable legal issues and financial regulations will be used to adapt services for producing and publishing video content in a global environment.
- Country-specific regulations might also impact other parts of the C2C processes. Besides the actual payment services, billing and distribution processes could also be constrained or adapted according to regulatory aspects.

Here we see context factors, which belong to all three levels of granularity, and are gathered using both automated and manual methods. Many of these features are shared with the other two case studies, and so is the fully distributed nature of context gathering and use for service adaptation.

2. State of the Art in Context Adaptation for Services

In this section we briefly review the existing work on using contextual information within Web Services research focussing both on conceptual and infrastructural research. The review is focussed on three main areas: the use of context to enhance Web Services; the existing context architectures and frameworks; and the different conceptual models for capturing context information proposed so far. We shall address them next in this very order.

2.1 Web Services and Context

Since the appearance of Web Services technologies and impelled by the global interest on Enterprise Application Integration [1] and Service-Oriented Computing [2], there has been much research and development devoted to better supporting the use of Web Services as the core constituent of distributed applications. Within these efforts much work has focussed on the definition of standards such as WSDL or the so-called WS-* family, whereas others have devoted their time to providing better execution frameworks and architectures [1].

Encompassing both standards and infrastructures has been the pressing need to achieve flexible solutions able to adapt to environment changes. Most of the research in this respect has focussed on handling exceptions and failures [3, 4], providing further security [1], and managing the Quality of Services (QoS) [1, 5, 6]. These approaches have typically been gathered under the umbrella of service monitoring and management and have mostly remained disconnected from research in context-aware systems, which has focussed mostly on the integration of mobile devices rather than services.

Recently however researchers within the context-aware area have been increasingly applying Web Services as an integration technology [7], and conversely, researchers focussing on Web Services related technologies are now beginning to consider aspects such as security, quality, trust and adaptability within the broader area of context-awareness [8-10]. In fact given the broad meaning of context, characteristics such as low-level execution monitoring data, previous services, security and trust concerns, legal regulations affecting service consumers and providers are valuable and even necessary contextual information concerning the services.

Semantic Web Services research [11-16] is based on formal models for both the data exchanged between services and the services themselves in order to support further automation of tasks such as the composition, mediation and discovery of services. Hence, Semantic Web Services are arguably context-aware to some extent since they resolve semantic (merely contextual) heterogeneities, and can adapt their behaviour to changing environmental conditions such as the user trust [17] or preferences in general [8].

The use of context information in general within Web Service-based environments is still however very much focussed on particular aspects and hence does not provide a systematic and generally applicable approach. Bringing further context-awareness to Web Services will indeed enhance the overall behaviour of the systems, and it is in fact regarded as a key feature to cater for a wider spread and application of Web Services [8].

2.2 Context Frameworks and Architectures

Particularly relevant to our work on adapting services to the context at hand is the research on developing context-aware applications. We understand by context-aware applications, those that make use of contextual information to adapt their behaviour in an attempt to better fulfil the task they are in charge of [18]. The development of context-aware applications, as it is nowadays widely understood, has mostly been undertaken within the areas of mobile and ubiquitous computing, and more recent trends such as pervasive computing and ambient intelligence. One could however argue with good reason that research in Artificial Intelligence

on signals interpretation [19], or even research in robotics [20], is extremely close to context-awareness.

Over time many different approaches to developing context-aware applications have been devised [7, 18, 21, 22]. In most cases, applications were developed in an ad hoc manner and thus tailored to specific environments, domains, and purposes. As a consequence, from a general perspective the solutions proposed embed a certain set of trade-offs that carry diverse advantages and drawbacks. What can be distilled from these applications is the complexity to design, develop and maintain context-aware applications. In the light of these difficulties, researchers have worked on defining frameworks and architectures for developing and supporting context-aware applications [7, 21, 22]. In [7] the author reflects on the diverse architectures and frameworks produced so far, and distinguishes three main approaches: widgets [22], the infrastructure model [23], and the blackboard model [7, 19].

The widgets approach adopts ideas from the architecture of GUIs for the development of context-aware systems. In this model, applications are built by incorporating widgets that are managed by a central component. Each widget provides a piece of context-adaptive user interface. Different kinds of widgets are contemplated, namely Interpreters, Aggregators, Services and a Discoverer. Context Interpreters support deriving higher-level contextual information from raw data. Aggregators collect distributed contextual information into a (virtual) coherent whole. Services provide context-aware pieces of functionality that can be reused in different environments. Finally, the Discoverer supports determining which functionalities are available for applications built on top of the framework.

The so-called infrastructure approach is a related but more flexible model that applies the Client/Server architecture. A key aspect of this approach is the complete distribution of the solution, which does not include a centralised component such as the widget manager. The infrastructure approach has been defined on top existing communication standards, which therefore enable their use in Web-based environments. Within this approach important efforts have been devoted to implementing discovery services that could support applications in finding and using deployed context-aware services.

Finally the blackboard model of developing context-aware applications is based on the model that originates from research in Artificial Intelligence during the late 70's and 80's [19]. In a nutshell the blackboard model of problem solving is based on the idea of having a shared workspace, the blackboard, which is used by collaborating agents as the sole communication medium. The blackboard captures the state of the problem and the experts contribute in an opportunistic manner when they see an opportunity. In this kind of context-aware applications, sensors store information in the shared blackboard, which is constantly monitored by other components in order to derive further information, or take context-driven actions.

In addition to these frameworks classification, Winograd [7] characterises the different approaches along a number of dimensions, namely efficiency, configurability, robustness and simplicity. The widgets approach appears to provide good efficiency and control, but reconfiguring or extending existing systems turns out to be somewhat complex and it lacks robustness. On the contrary, the infrastructure solution is less efficient but it enhances the configurability and robustness of the systems. Finally, the blackboard approach provides even more configurability through a greater decoupling of systems, its high modularity simplifies the construction and understanding of systems, and it provides further robustness since the data is constantly available for all the components. The only drawback Winograd identifies in this case regards the communication overhead since every interaction between two components involves an additional indirection. However, an aspect that the author does not take into account is the fact that communications might be between one component and many others in which case that additional overhead can safely be neglected.

2.3 Context Modelling

A key aspect of context-aware systems is concerned with the modelling of contextual information. A variety of models have been proposed so far. Early versions focussed on application-specific models whereas the wider application of context in different scenarios brought the interest of defining general-purpose models able to capture contextual information for diverse applications. In [24] the authors present a survey on existing modelling techniques. The study is carried out in the light of six fundamental aspects that are considered of utmost importance for modelling context, i.e., distributed composition, partial validation, richness and quality of information, incompleteness and ambiguity, level of formality, and the applicability to existing environments.

Guided by these factors, [24] analyses the most relevant context models which the authors organise into six modelling approaches, namely key-value, mark-up scheme, graphical, logic-based, object-oriented, and ontology-based models. The main conclusion reached by the authors is that ontologies appear to be the most promising approach to modelling contextual information. In particular, the modelling facilities provided are considered as intuitive as the object-oriented approach but still remain formal, which is the main advantage of logic-based models. Ontologies provide a uniform way for specifying concepts, instances, relationships, properties and even axioms in a way that can be understood by humans, and more importantly, processed by machines. Ontology-based context models support comparing contextual facts, deriving more complex contextual information from base measurements, and most importantly they are sharable representations, which makes them suitable for distributed, and heterogeneous settings.

In [25] an extension of the previous survey focussing solely in the existing context ontologies is presented. The authors introduce a set of context modelling criteria, which on the one hand establish common requirements for context models, and on the other hand provide several dimensions for comparing existing models. The criteria are the *applicability* to different domains and situations; the *comparability* of values in different units and the support for non numerical values; the *traceability* of contextual information for determining its provenance and thus its reliability; the support for *logging* or keeping historical records of context values; means for capturing the *quality* of the contextual data; the capacity for assessing the *satisfiability* of context values with respect to prescribed models; the capacity for *inferencing* or deriving higher-order context knowledge; the capacity for dealing with *incompleteness and ambiguity.* In addition to these context modelling-specific criteria, the authors take into consideration quality criteria for ontologies in general [26, 27], such as reusability, extensibility, genericity, consistency, completeness, etc.

From a genericity and applicability perspective several context ontologies appear to be quite appealing. We can cite in this respect ConOnto [28], ASC [29], mySAM [30], GUMO [31], and SOUPA [32], to name a few. The latter however is probably the most renowned upper ontology for modelling context. Most of the models deal with the quality of the information, however aspects like comparability and traceability are often neglected. With respect to supporting comparing contextual information expressed in different units ASC and GUMO are probably the most suitable options. Concerning traceability CONON [33] and the Context Management Framework of VTT Finland [21] are the only ones, to the best of our knowledge, that provide direct support.

In general terms what can be distilled from the state of the art in context modelling is the plethora of proposals that unfortunately do not provide a full coverage for all the desirable features previously introduced. In fairness to the respective authors it is worth noting that certain features are in some cases partly covered architecturally. The reader is referred to [21, 24, 25, 28, 30, 32-37] for further details.

3. Overall Vision on Contextual Service Adaptation

Service Oriented Architecture is increasingly being adopted as the underlying architecture within and between enterprises for a variety of reasons at a technical and business level. On the technical level its suitability as an Enterprise Application Integration paradigm is perhaps of most relevance. From a business perspective, considering applications as services has important advantages since it reduces the gap between current practices within the business world and those adopted in the IT domain.

SOA4All will contribute further to the uptake of service-oriented technologies in order to realise a world where billions of parties will expose and consume services via the Web. The scale of the vision pursued by SOA4All will however bring further heterogeneity to SOA approaches that we need to accommodate and deal with at the infrastructure level. One such aspect that is increasingly seen as a *condicio sine qua non* for achieving this vision is the capacity to dynamically adapt services based on contextual factors. These factors range from immediate concerns of location and language to legal issues and financial regulations. Swiftly accommodating to the context at hand will become increasingly important as their diversity expands with the global reach of the future Web of Services.

We previously saw that many different approaches to developing context-aware applications have been devised so far. The main conclusion that can be drawn from these applications is the complexity to design, develop and maintain context-aware applications. These difficulties range from purely technical difficulties to higher-level conceptual concerns. In a nutshell, bringing context-awareness to Web Services is regarded as a key feature to cater for a wider spread and application of Web Services [8], but doing so in a systematic manner necessarily requires i) a fully-fledged general purpose framework designed to provide an efficient, configurable, robust and still simple solution to context adaptation; and ii) appropriate means for modelling contextual information based on a set of necessary characteristics like the applicability of the model, the support for comparing data, or the ability for inferring or new data.

The Contextual Service Adaptation Framework described herein will provide a generic platform for supporting the adaptation of services to diverse contexts according to a virtually infinite variety of dimensions. In fact, our understanding of context is very much in line with perhaps the most widely agreed definition which is presented in [18]:

> "*Context is any information that can be used to characterize the situation of an entity. An entity is a person, place or object that is considered relevant to the interaction between a user and an application, including the user and the application themselves*."

In SOA4All we do however take a slightly wider approach to context that is not limited to interactions between users and machines but also between machines. After all, one of the main advantages service-oriented technologies provide is the capacity to compose existing services in order to provide more advanced ones.

Somewhat implicit in this definition of context, is the fact that the relevance of contextual information is dependent on the actual application or service being provided. For instance, the location of the consumer might be irrelevant for some services whereas it might be key in other situations where for example the appropriate legal regulations need to be applied. In fact the underlying essence is that what can be considered as relevant contextual information depends on the task being performed. In a scenario where billions of services are provided and consumed, the diversity of the tasks that will be supported will presumably be outstanding and there needs to be appropriate means for supporting adapting services to contextual factors rather than providing specific services for each and every particular context one may encounter.

In the light of this broad understanding of context, our approach to context adaptation is based on a set of conceptualisations providing the means for modelling contextual information of any kind, and a general purpose machinery able to manage contextual data, use it for recognising concrete contexts within particular situations, and apply this contextual knowledge for adapting services. Therefore, central to our approach, much like the definition of context itself, is the genericity of our framework so that it can cover the wide range of situations one is likely to encounter in a Web of billions of services.

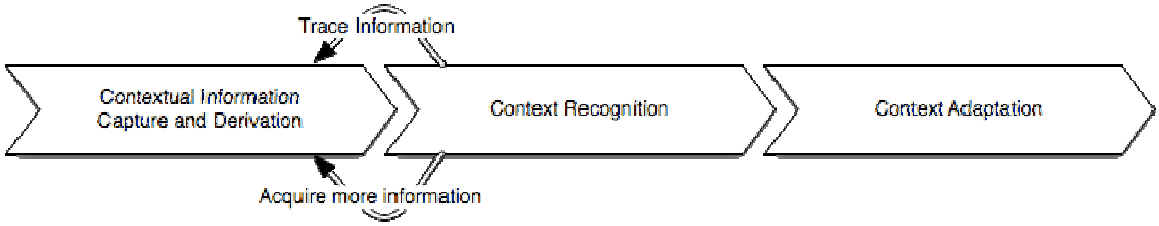

Figure 1. Contextual Service Adaptation Life-Cycle.

Adapting services to particular contexts is envisaged as a process like the one illustrated in Figure 1. In a nutshell, this process involves gathering and deriving contextual information, recognising a certain set of relevant contexts given a concrete situation, and eventually adapting the execution of the service based on the contextual knowledge available. Although this process is essentially sequential, while trying to recognise contexts it might be required to acquire more information either from raw data or by deriving additional data on demand, and to trace the information obtained in order to determine how trustworthy it is.

Our framework will provide three main generic services to the overall SOA4All vision: i) Context Management; ii) Context Recognition; and iii) Service Adaptation based on Context. We shall base our development on the use of previous research in the area of Knowledge Engineering and Artificial Intelligence for capturing the expertise for supporting Context Recognition and Service Adaptation in a domain-independent way, and ontologies both for defining the interfaces of these services and for capturing contextual information in a sharable and machine-processable way.

Context Management will be supported in our framework by means of a set of ontologies that will provide an appropriate basis [24] for formally capturing contextual data. Contextual information will be gathered both at runtime through monitoring the execution of services as well as at design time based on user interactions or more advanced analysis of previously gathered information (e.g., computation of high-level performance indicators, clustering, etc). This context information will be managed by a federation of Context Repositories forming a cloud of distributed data, which will profit from the global scale of the infrastructure, to derive further information from emerging social networks in a Web 2.0 fashion. We foresee that this constellation of Context Repositories will give birth to new business models centred on the strategic business value of contextual information gathered over time by diverse organisations within particular communities.

The other two infrastructural services, namely Context Recognition and Service Adaptation, will be based on so-called Problem-Solving Methods (PSM). Many researchers have advocated the development of Knowledge-Based Systems by reusing Problem-Solving Methods (PSMs), that is software components that encode domain-independent sequences of inference steps for solving particular tasks [38-43]. PSMs provide access to their functionality by means of formally defined interfaces (e.g., ontologies) in a domain-independent manner so that their expertise can be reused across different domains. For example, a Configuration PSM defines the concepts and relationships required for solving any kind of configuration problem, independently of the domain it is applied to, e.g., elevators construction.

Our epistemological basis for context information derivation and context recognition is based on Heuristic Classification [44], a well-known PSM [38] for solving Classification tasks [45]. In a nutshell Heuristic Classification is a Knowledge Level [46] method that "*relates data to a pre-enumerated set of solutions by abstraction, heuristic association, and refinement*" [44]. Within our framework we therefore approach the task of determining a (set of) relevant context(s) based on a wide range of information concerning users and services, as a process involving the abstraction of information gathered, its heuristic classification with respect to general cases, and the refinement to the particular case at hand. In fact, it is this very process of heuristically classifying contextual information that produces contextual knowledge which can then support the appropriate adaptation of services. The reader should notice that we are here distinguishing between *contextual information* (i.e., all the contextual information gathered may it be useful in the situation at hand or not) and *contextual knowledge* which can support the system acting rationally [46].

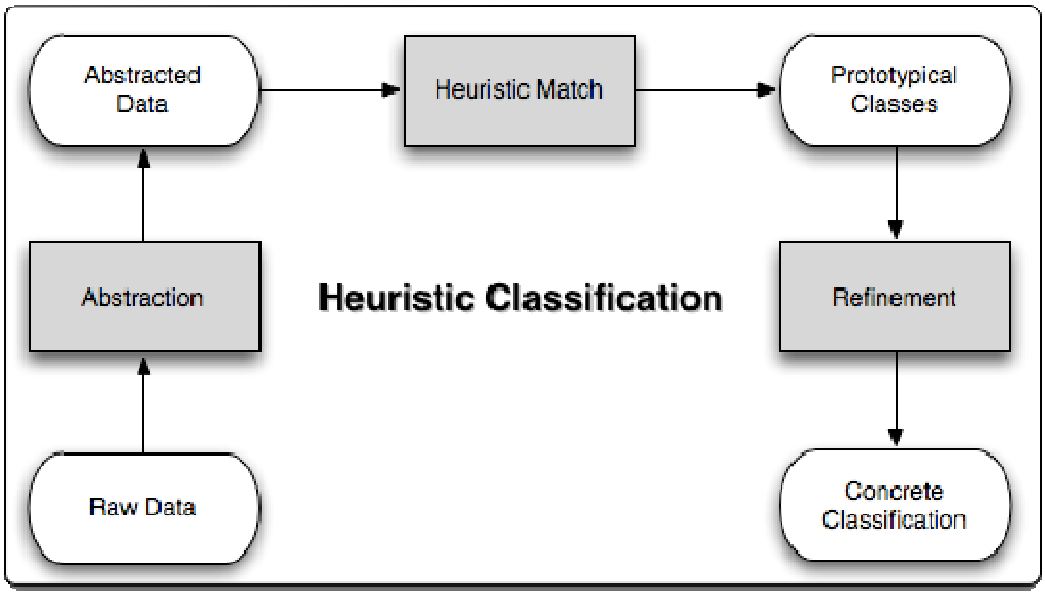

Figure 2. Heuristic Classification.

Once the particular context or sets of contexts have been determined, our framework will support the adaptation of services at design time, while creating composite services, and at runtime based on available contextual information at that particular moment. The main concern in both scenarios is that services that are *a priori* compatible might not used together within certain contexts (e.g., providing an Internet connection of a certain quality might not be possible depending on the type connection or device used). Again, we build upon the use of a generic Problem Solving Method to support this task. In particular, we will use Parametric Design [40]. Parametric Design is a simplification of Configuration Design [45], whereby the components to be configured have a parameterised solution template to guide the design process. In this particular scenario design problem solving is a matter of assigning values to these parameters. In a similar vein to that presented in [47], we view the adaptation of services to particular contexts as a Parametric Design task where the parameters are context dependent features of the service such as the currency, the kind of user, etc. The main idea is therefore to minimize the computation complexity of typical approaches such as Planning by providing templates and supporting customization through Parametric Design.

Based on the elements of the framework delineated above (ontology stack, classification and parameterization mechanisms), the overall context adaptation mechanism would run as follows. The ontology stack is built in establishing key parameters shaping contexts. The

values of these parameters (recorded in repositories) can then be used to adapt the services offered to a user in commensurate terms (for example, in tailoring the language of the interface and other accessibility requirements). There are at this stage two potential levels of context adaptation: i) user-driven customisation and ii) automatic adaptation. As a middle term between complete manual set up and fully automatic adaptation, user-driven customization could be supported by "intelligent customisation wizard" (software-based assistants). The introduction of such tools would enable a seamless continuum of context-driven adaptation of services (from non-automated to fully automated).

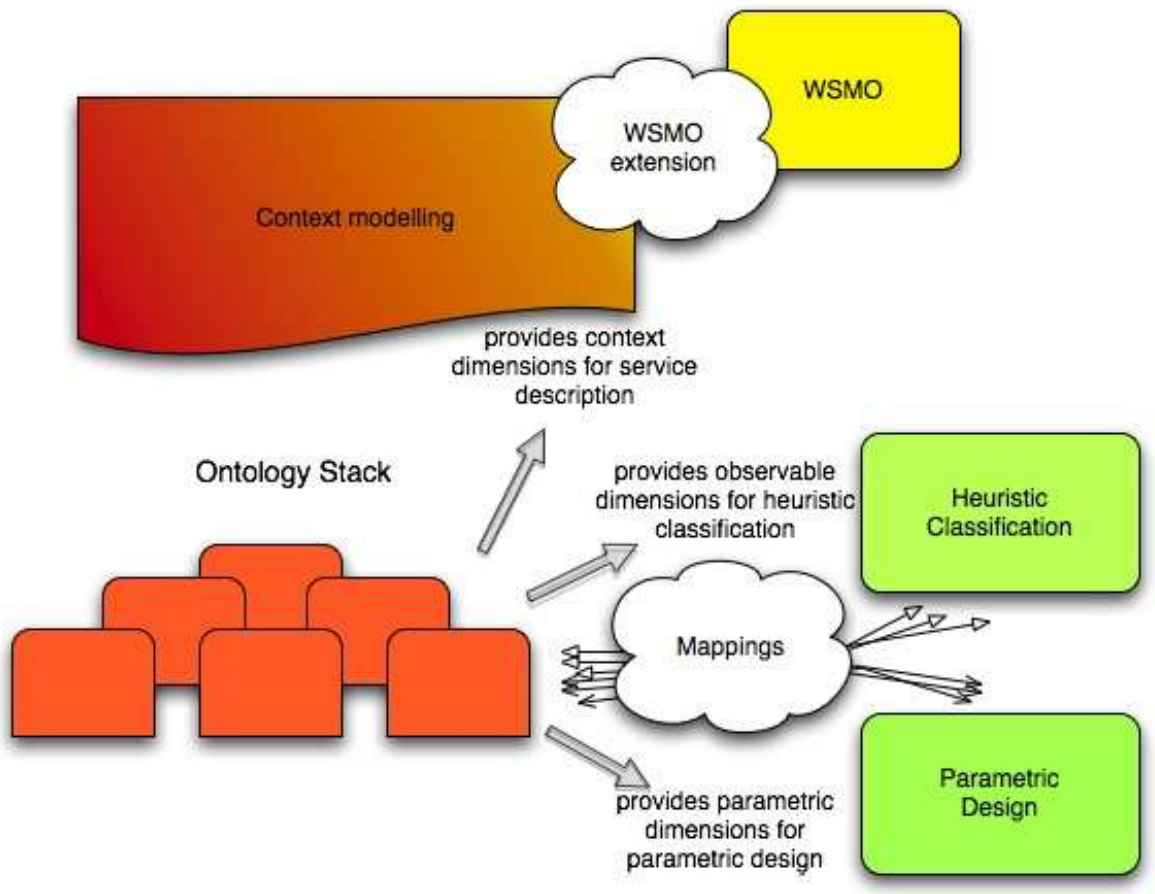

Figure 3. Overview of the Contextual Service Adaptation with Emphasis on the Role and Integration of the Ontology Stack.

Support for service adaptation will thus be realized through the articulation of WSMO-based descriptions extended through the use of a number of context modelling ontologies and the application of Heuristic Classification and Parametric Design Problem-Solving Methods geared toward the processing of contextual information as illustrated in Figure 1. To this end, the appropriate mappings between the ontology stack and the other elements should be defined so as to support the use of all sorts of contextual information within both PSMs. In this way contextual information will be treated in one case as observables that help to characterise a certain context, and in the other as parameters for services. Section 6.2 and 6.3 provide more background on the methods alluded to here. At this stage, the overall ontological treatment of context modelling can be tentatively described as constituted of the following:

- ontology stack with its proper degree of integration within its components,
- mappings to service descriptions, fundamentally building on WSMO and its

prospective extensions,

- mappings between the ontology stack and the relevant elements in the classification and adaptation mechanisms.

In the remainder of the deliverable we shall provide a more detailed description of the Contextual Service Adaptation Framework, introducing the main components involved, and outlining their interactions and their internal technical details. The reader should however bear in mind that this document is not intended to provide a fully detailed solution but rather an overall design for guiding future research and development.

4. Contextual Service Adaptation Framework

In this section we focus on deeper technical details showing how we foresee the vision previously outlined can be achieved mentioning the possible or in some cases expected technologies that will be used. Figure 4 shows the overall architecture envisaged for the Contextual Service Adaptation Framework. The framework provides three infrastructural services, namely Context Management, Context Recognition and Context Adaptation, which will be provided both jointly as a global infrastructural functionality able to support adapting services based on a particular scenario, as well as separately so that applications can make use of these three infrastructural services as they consider more appropriate.

This decoupling brings the necessary modularity and flexibility for its application within different applications and even within different components of the SOA4All architecture. In fact, although fully-fledged applications might want to make use of the overall framework as a single entry point service in order to entirely automate the adaptation of services, other applications and components of the SOA4All architecture might want to use discrete pieces of functionality at will. For instance, the Service Provisioning Platform devised in WP2, will be able to make use of the Context Recognition service in order to adapt the User Interface based on the user expertise and preferences, and it will be able to order services at discovery time based on derived profiles, without having to make use of the Context Adaptation service.

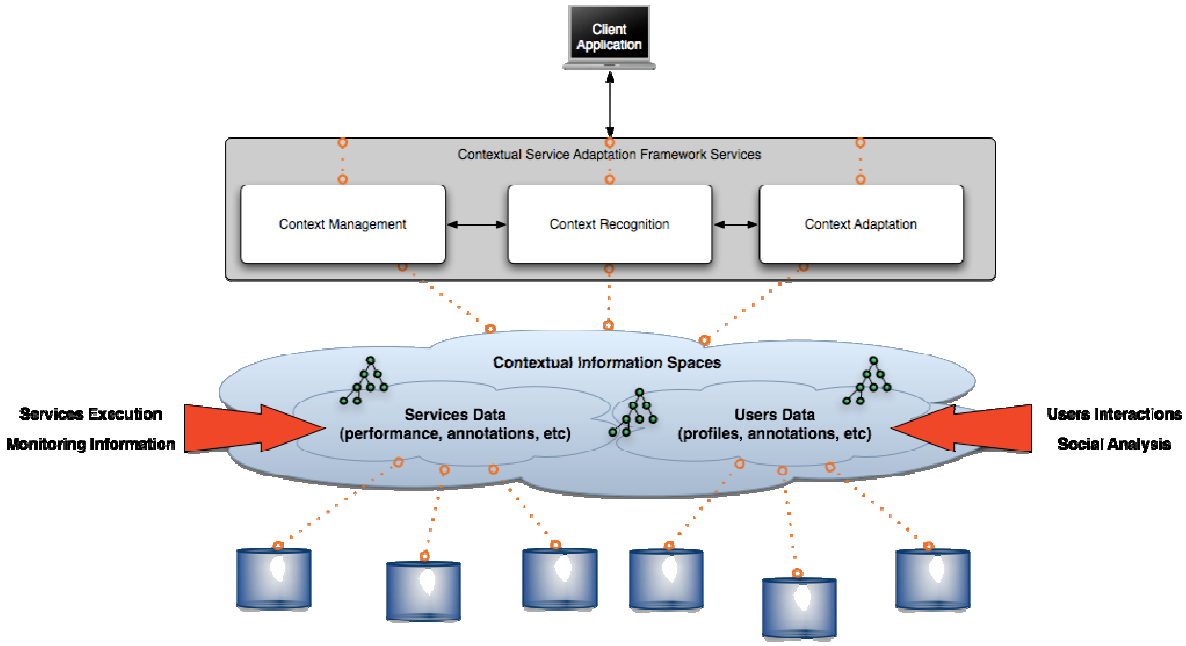

Figure 4. Contextual Service Adaptation Framework Architecture.

In the remainder of this section we shall cover the main components of our framework, starting first with the Context Management Service, moving afterwards into the Context Recognition Service and we conclude with the Context Adaptation Service.

4.1 Context Management Service

As previously introduced, the overall framework will be supported by a cloud of contextual information, see Figure 4. Given the goal and scope of SOA4All, that is a web of billions of services, a vast amount of contextual information will come from a distributed and heterogeneous set of sources that need however to be integrated in a seamless, coherent, and efficient manner. To cater for this, the framework will be supported by a Context

Management service which will be in charge of the following activities [48]:

Context Information Representation: This service will provide the appropriate means for capturing contextual information in a formal way, allowing the application of knowledge-based techniques for recognising contexts and adapting services, but also in a flexible way to cater for a virtually infinite number of domains;

Context Information Storage and Querying: The Context Management service will support storing large amounts of contextual information expressed in terms of the context ontologies in a distributed, integrated and efficient way. Access to this information should be provided in a flexible and efficient way supporting the retrieval of relevant information on demand as well as the automated notification when important information has been added;

Context Abstraction and Interpretation: Contextual information can be divided into raw data generated directly by users' interactions (e.g., preferences), or services execution (e.g., execution time), and higher-level data that will be derived (abstracted in heuristic classification terms) automatically based on more advanced analysis techniques (e.g., Quality of Service). The Context Management service should support the application of abstraction techniques able to derive higher-level contextual information out of raw data captured by the infrastructure. Indeed, the means and frequency for deriving contextual information will depend on the domain being processed. For instance some contextual information, such as the information about applicable legal regulations and business registration processes in the WP7 case study, will be fairly static, whilst other pieces of context information, e.g., information about service execution times, will be highly dynamic. It is therefore important, as we previously outlined in Figure 1, to support triggering this computation on demand to reduce the overhead associated.

In the reminder of this section we shall cover in more detail each of these functionalities outlining where possible the techniques an approaches we envisage will be applied within the framework.

4.1.1 Context Information Representation

As indicated before, the representation of context information will rely on the descriptive resources of a number of ontologies that will be used, in particular, to enrich the description of services based on WSMO and its prospective extensions. The ontologies on the basis of which contextual information will be added form together the ontology stack. We will draw on the requirements identified in [25] as key factors in ensuring a successful implementation of such an ontology stack; these requirements are as follows:

Applicability: We are interested in supporting a range of applications and the context modelling mechanism has to be broad enough to allow for them. Pitfalls would occur if the context ontology stack imposed too strong restrictions on the range of services that it can actually be used to support. For example, if a model lacks the ability to handle temporal aspects, it cannot be used to support applications dependent on temporal information.

Comparability: The services we intend to support typically use attributes whose values can be compared and ordered. For example, monetary values (e.g. prices) or durations (e.g. delivery times). As there may be different systems of units (currency, calendar units) we need to be able to handle conversion and comparisons. We can make provision for expanding this requirement to features whose comparison is more qualitative than quantitative, such as when comparing the contextual environment in which a service is invoked (the user is in her office ordering a book as part of a working task or she is at home ordering a book as a

Christmas gift) or more simply when the user is in a bad or good mood (indicating, for example, her tolerance to advertising niceties).

Traceability: Provisions need to be made for supporting identification of information sources in order to evaluate reliability. For example, a fact about a user preference may be given greater or lesser weight whether it has been entered by the user in a dedicated form, perhaps even specifically when interacting with the service provider, or whether it's been gathered from a public web forum.

History, logging: This is related to traceability but emphasizes the temporal factor in the evaluation of the reliability of a source. Indeed, the information provided may be dated and less valued than a recent update. We can make provision for supporting the recognition of key events such as, for example, registration with a service provider and last login as well as taking into account and checking changing data (updated user information).

Quality: Further towards supporting the reliability of contextual information and the framework processing it, the mechanism has to allow for the evaluation of the quality of information it processes. Moreover, it needs to be able to discriminate between pieces of information that are not on a par according to suitable evaluation criteria. Joining up with the traceability requirement, a mechanism for evaluating the reliability of sources of information () can also be envisioned linking.

Satisfiability: Some degree of quality assurance has to be enforceable on contextual models themselves. A mechanism has to allow for the assessment of whether a context instance satisfies a related set of constraints.

Inference: Inferencing, that is to say the ability to execute derivation mechanisms or means, needs to be supported so as to allow for the identification and specification of high-order context knowledge, for example in the form of context abstract or classification, on the basis of low-level context information.

Dealing with Incompleteness and ambiguity: Contextual information is usually noisy and incomplete. This pervasive and difficult issue can be addressed to some extent using some of the features of the mechanisms that would satisfy the previous requirements. For example, if the satisfiability requirement is met through the use of a number of constraints exerted on the information processed, values outside permissible ranges could be discarded as noise.

Whilst the foregoing set of criteria can be seen as addressing the overall modelling facilities provided by the ontology-based mechanism for the description and specification of context for services, the following set can be used as a basis for evaluating the actual ontologies produced and guiding the process of ontology engineering leading to them.

Reusability, standardization: Reusability is a measure of the number and variety of modelling tasks an ontology can be used to support. Standardization is not an immediate correlate but the greater the reusability, the more standard an ontology may be in relation to its application space. In short, we are talking about a measure of the applicability of an ontology.

Flexibility, extensibility: Flexibility and extensibility, are measures of how easy or productive it is to expand an ontology. This may involve adding new concepts or constraints and dependencies while preserving the original bit of the ontology.

Genericity: Genericity relates to reusability (hence to applicability as well). A high level of genericity is a defining of those ontologies sometimes called upper-level, that is, ontologies containing elements reusable in many different domains. Genericity is however an absolute concept (an ontology is generic or it is not) while upper-

levelness is relative. There are nevertheless degrees of genericity (and specificity).

Granularity: This relates to the diversity and coverage of individual concepts. The granularity of an ontology in this sense may be correlated to its genericity as upper-level ontologies tend to be of a coarse granularity. Again if we consider that being on top of something is a relational matter, we see that we can nevertheless make provisions for rather fine grained upper-level ontologies. The desired level of granularity for an ontology is often dependent on the context of application and development. Thus, in the context of SOA4All, a measure of the desired level of completeness will be tentatively indicated by the use cases requirements.

Consistency: A consistent ontology is one that does not contain any contradiction. This is a trivial requirement as an inconsistent ontology is just an example of a bad ontology, i.e. one that should not be.

Completeness: Completeness is a measure of the coverage provided by an ontology of a given domain. Ideally, an ontology is complete, however, depending on the granularity that the ontology is intended to achieve, achieving completeness may be very difficult and not always desirable. The stake is to identify the *right* level of completion for an ontology. As with granularity, this is often dependent on the context of development. Thus, in the context of SOA4All, a measure of the desired level of completeness will be tentatively indicated by the use cases requirements.

Non-Redundancy: Non-redundancy indicates the presence of duplicated concepts or relationships, with different names yet sharing overlapping formal definitions. Redundancy comes across as the dual of inconsistency and is as undesirable.

Readability: A readable ontology contains intuitive names for concepts, so it is easy to understand and adopt. Although this may come across as a nicety, especially given the fact that ontologies are intended to be processed by machines and not human beings with intuitions, experience shows that such provision influence development (affecting both consistency and non-redundancy when human editors are involved) but moreover reuse and adaptation to distinct application contexts.

Scalability: Scalability indicates to which extent and degree it is possible to implement processing of the specification of an ontology (e.g. how many concepts and relationships and at what cost).

Language, formalism: Ontologies can be specified in a variety of knowledge representation languages supported by more or less powerful logics and associated inferential mechanisms. Choices in this respect may influence for some part the ability to meet other requirements, e.g. flexibility or granularity, but more importantly scalability in the sense just described.

The rough evaluation of available ontologies and the modelling mechanisms provided by them in terms of representing context information we carried in Section 2.3 shows that these efforts in context modelling fail to exemplify all requirements listed above. Ontology efforts within the Contextual Service Adaptation Framework will be devoted to providing a fully-fledged contextual modelling framework that aims at meeting these requirements. The second list of requirements strongly suggests that the solution has to come from the development of a set of articulated complementary ontologies that can be placed along the dimension of genericity and granularity and that exhibit a number of intrinsic features which have grown to be recognized as the result of best practice in ontology development (extensibility, consistency, and readability, in particular). This ontology stack will thus incorporate a number of sharply delineated components. Some of these ontologies will have

a relatively high-degree of genericity and constitute an upper-level for the other. Others, which will be more domain-specific, will have finer granularity and support the most specific aspects of context information as required by the use cases.

According to [49] the following aspects are of greatest importance when capturing contextual information:

- **Informational aspect**: relevant resources accessed or created by the user during a task but also the relevant subject or corresponding domains or topics associated to a task.
- **Organizational aspect**: organizational structures involved into or relevant to the task: person(s), roles, skills, interests of a user, projects and organizational units she belongs to.
- **Behavioural aspect**: the actual behaviour of the user – her performed operations and actions.
- **Operational aspect**: applications and tools used by the user (to accomplish her task).
- **Causal aspect**: task goals and estimated user goals.
- **Chronological aspect**: timeline of events occurred in the system, e.g., recently processed workflow tasks.
- **Environmental aspect**: the physical location but also the IT hardware used and the people present.

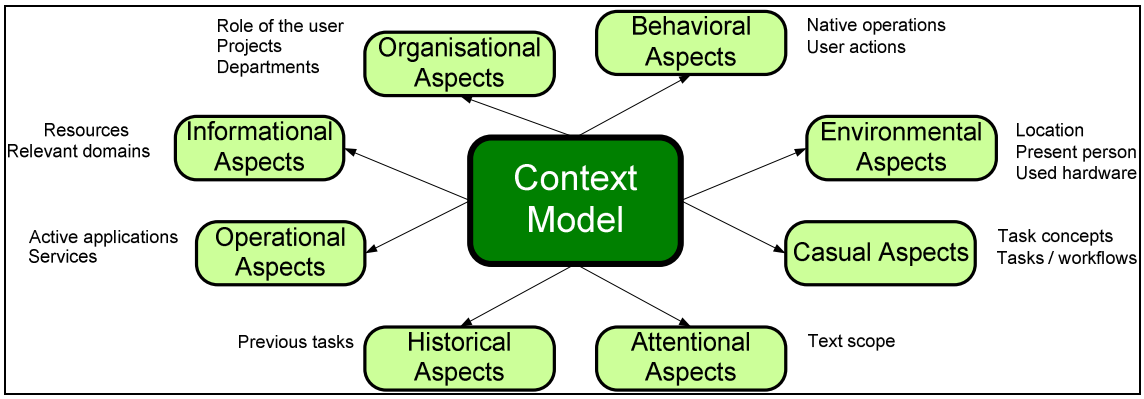

Figure 5. Aspects of contextual information.

In the context of SOA4All, core dimensions are those essentially concerned with the environment, the software and the actors involved in the consumption of a service. We may organize the different kinds of contextual information that we can foresee as relevant in that respect according to how they relate to one of the following:

The Task the actor is pursuing by consuming (invoking) a particular service (task horizon), a part of the so-called *causal aspects* of context;

The Service that is to be consumed (invoked), including information about its speed, reliability, etc., a part of the *operational aspects* of context;

The Actor (stakeholder) who is consuming (invoking the service), including their language and other skills, background knowledge, and general aims. This is a part of the *behavioural and organisational aspects* of context;

The Software and IT Environment in which the service consumption takes place, this includes factors such as operating system, available hardware and communication links (speed and cost), this is a part of the *environmental and operational aspects* of context;

The Physical Environment in which the service consumption takes place. This includes information about geographical location, time of the day, level of lighting, noise levels, and is the traditional focus for context adaptation pursued by the pervasive computing community. This is a part of the *environmental aspects* of context.

The Organisational Environment in which the service consumption takes place. This includes the role of the actor consuming the service (the set of their organisational rights and obligations), the norms governing the relationship of this role with the rest of the organisation, the organisational structures and policies. The granularity of organisation can by itself vary from SME through a department and a big multinational company to a country. This is a part of the *organisational aspects* of context.

The Societal Environment in which the service consumption takes place. This would include customs, moral norms and expectations. In effect the context factors are similar to the ones under the Organisational Environment but likely to be more flexible and fuzzy. This is also a part of the *organisational aspects* of context.

The specialisations of each of these kinds are largely dependent on the application context and as such they will be developed in detail after analysing the case studies of SOA4All.

For the sake of illustration, we will merely indicate here a number of tentative and prospectively useful types of context-related information pertaining to personal context. Echoing some of the examples used earlier and drawing also on [50-52], a list of typical features making up personal context could include:

- Goals and Motivation, including personal needs for services;
- Skills and Ability, including relevant knowledge and capabilities as well as general cognitive abilities, linguistic skills, literacy;
- Accessibility requirements and preferences in interaction (font size, preferred language);
- Connections (professional contacts or friends) and communicating modalities within a given network;
- History of services and record of preferences (‘Favourites’);
- Regular activity or activities (professional, hobbies, interests);
- Service level authorization.

Information of such kinds is not equally available, if at all, depending on the kind in question. To address the issue of knowledge acquisition, it is expected that the majority of the information gathered will come from historical data. This data will have to be interpreted in terms of context parameters. Other elements of personal contexts will have to be specified by the users themselves, possibly answering prompts by the system. Finally, Web2.0-style forums such as, for example, Facebook will be considered for they are readily available sources of potentially relevant data.

The way contextual features, those corresponding to the personal context among other, will have to be determined as relevant may vary depending on the application the context mechanism is intended to support. Indeed, it may even depend on the whole target domain of an application which elements ought to be included in the basic contextual features

supporting the mechanism. Given the context of development of the ontology stack, it would be ludicrous to attempt an equally detailed support for all possible aspects of contextual information. Rather the generic elements in the ontology stack need to be large sweeping enough so as to allow for opportunistic specification driven on the overall project rationale. This means in particular that, in the context of the project, the development of the ontology stack will have to be receptive to the requirements drafted for use cases.

From a more technical perspective the concrete representation formalism to be used will play an important role in terms of the expressivity that can be obtained, as well as it will affect the performance and the availability of software for manipulating the information. The knowledge Knowledge-Based Systems build upon is typically characterised as static or dynamic. Static knowledge captures the concepts, properties, and relationships for domains of interest (e.g., medical, manufacturing, etc.). Static knowledge is most often represented with ontologies, which are widely understood as "formal explicit specifications of a shared conceptualisation for a domain of interest" [53]. Dynamic knowledge is usually defined as a mixture between traditional software procedures and inference rules that declaratively define reasoning steps. Indeed, the ability to define behavioural aspects when developing Knowledge-Based Systems is important, and in many cases essential, as Musen points out in [54]: "Many intelligent systems are designed primarily to answer queries about large bodies of knowledge. In these cases, ontologies provide most of the representational muscle needed to build complete systems. To build systems that solve real-world tasks, however, we must not only specify our conceptualisations, but also clarify how problem-solving ideally will occur."

Although at this early stage in the project important aspects of the architecture are still unspecified and we cannot therefore categorically establish the language we will use. However, we can already advance the need for representing large amounts of static knowledge (i.e., context information) as well as the necessity to capture advanced dynamic knowledge (i.e., procedural problem-solving knowledge) capable of supporting the recognition of contexts (i.e., heuristic classification) and the adaptation of services (i.e., parametric design).

On the one hand, given that the project is essentially Web-oriented, static knowledge will be represented using Web standard languages for representing ontologies (e.g., RDF/RDFS [55] or OWL [56]). The trade-off between expressivity and performance (or even tractability) will need to be carefully analysed as soon as more detailed information about the particular use cases will be available. It is however expected that RDF/RDFS will provide a sufficient basis for capturing distributed information in an efficient way and the lack of expressivity will be compensated by the other infrastructural services of the Contextual Service Adaptation Framework. On the other hand, dynamic knowledge for supporting the recognition of contexts and the adaptation of services will require the capacity to express problem-solving procedures as supported by languages like WSML [14], OCML [57] or even traditional programming languages such as Java.

4.1.2 Context Information Storage and Querying

Context information will come from a plethora of heterogeneous and distributed sources and will be gathered and generated in different ways. This poses therefore an important number of challenges that need to be addressed appropriately. In the reminder of this section we cover the requirements we have identified and we will introduce the technology we envisage to apply to cater for this.

4.1.2.1 Requirements

The technical requirements for the storage and querying of contextual information will mainly depend on the expressivity of the language used, the amount of information to be manipulated, the location of the data, and its dynamicity. Among the different kinds of

information we contemplate are:

- **Application-specific context information**: This information will be relatively small, domain specific and mostly static in nature.
- **User provided data**: Users will contribute to enhance the annotations of services by tagging them, reviewing them, recommending them, etc. This type of information will be provided in an incremental manner and relatively slowly. The sources for this data will be distributed over the Web.
- **User monitoring information**: This information will track user preferences such as the language, the preferred level of detail, the geographic location. This information will be gathered automatically by the infrastructure while the user provides and consumes services. This information will be slightly larger in volume, it will be captured in a fully distributed manner, and will be dynamic.
- **Services monitoring information**: This information will be generated automatically by the service monitoring infrastructure. It will therefore be highly dynamic, of a massive scale, and captured in a fully distributed computing environment. Depending on the kind of application we might require a more or less timely notification of changes in the context.
- **Derived high-level information:** Based on the raw context information gathered over time, the Contextual Service Adaptation Framework will process and derive higher-level context information. For instance Web 2.0 techniques will be applied in order to perform global profiling of users in order to derive preferences, or complex performance indicators will be computed over raw services monitoring information etc. This information will be small and slightly dynamic since it is not expected to be recomputed often. The main requirement stemming from this type of information concerns the need for processing large amounts of previously gathered data in batch processes.

These main types of sources of data allow us to derive the following main requirements the context repository should fulfil:

- **Distributed storage and retrieval:** Most of the data gathered about context will be generated in a distributed fashion. However, the overall body of contextual information should be accessible as a whole independently from its location.
- **Massive scale:** The data at the lower levels of granularity will be generated at very fast rates, for example monitoring information about services will be updated every time a service is executed, regardless of the place of execution. On the one hand this makes any storage system based on centralised solutions quite impractical. On the other hand the amount of data to be manipulated is expected to be very large.
- **Knowledge based:** The data gathered by context generation mechanisms will be expressed in terms of the context ontologies and should support reasoning in order to derive higher-level context information and support advanced querying facilities.
- **Event-based support:** Given the fast pace by which data will be added and updated, there is a clear need for supporting an event-based notification of interested clients upon reception of certain data. Agents should be able to subscribe to certain kinds or patterns of data so that they are timely notified whenever new relevant contextual information is available. This functionality is envisaged to be of major relevance for WP8 use case whereby methods for ensuring that certain Service Level Agreements are met should be put in place.

4.1.2.2 Semantic Spaces as Context Repository

The requirements exposed above are indeed particularly challenging. WP1 proposes the use of Semantic Spaces "as the underlying infrastructure for supporting the storage, querying and reasoning about distributed information efficiently over the Web". Semantic Spaces provides an architectural layer in the SOA4All architecture, used for both storage and communication. Semantic Spaces will be used in SOA4All for storing semantic descriptions of services (i.e. WSMO, SAWSDL and WSMO-Lite), monitoring data of service execution (i.e., performance and scalability properties) as well as service compositional information (i.e., business processes based on semantic descriptions of services for service composition).

Semantic Spaces are envisioned to be based on a large set of repositories physically distributed but connected to each other forming a virtually single shared storage space. Semantic Spaces support the storage of RDF information, the efficient retrieval of data by means of the SPARQL query language. In terms of communication, the Semantic Spaces approach can be used to coordinate the views of the different components in SOA4All. Semantic Spaces is based on principles from tuple-spaces, and does therefore also provide the means for subscribing to certain kinds of information patterns so that relevant information is distributed to the interested software in an efficient and automated way. Thus, it represents a means for supporting the effective coordination of different software components in a proactive way much like the so-called blackboard model. This type of architecture, as we previously saw in the state of the art section, provides the required reactivity, configurability, extensibility and complete decoupling as necessary for context adaptation.

In short, Semantic Spaces seems to address the generic requirements formulated above and therefore represents a particularly appealing candidate for storing contextual information in a distributed, scalable, and yet efficiently retrievable way. Additionally it provides support for an appropriate model of coordination based on the blackboard paradigm. There remains however the need to explore the reasoning capabilities provided and the well-known trade-off between the expressivity and the efficiency of the platform. Within WP4 efforts will be devoted to analysing the capacities of Semantic Spaces as the underlying infrastructure for storing Contextual Information and if necessary specific enhancements or different solutions will be contemplated.

4.1.3 Context Abstraction and Interpretation

As we previously introduced, contextual factors range from immediate concerns about location and language to legal and financial issues. Beyond those factors, an important issue in context information is how such information can be generated and deployed according user interaction in the creation and composition of services and across different executions. Context is an important source of information that can be used to adapt services to make them more suitable to user needs, but should also support a system to suggest services based on profiles similarities between users context. This could help to increase the level of knowledge and expressivity among users, as long as it is aligned with project's goals helping to create a useful and collaborative community of users.

In SOA4All, context information will be used to provide relevant information during the composition and execution of services, during the discovery of services, and it will be used to adapt the user interface of applications, etc. The relevance of such information depends, first and foremost on the task at hand, but also on the user preferences and expectations, his past behaviour and the particular services involved [18]. This is due to the fact that services may be influenced by additional information that is defined beyond the application logic and is instead dependent on contextual factors. The ability to adapt those services in terms of the context becomes very important [58] to reduce the amount of required user interaction [35]

and assure users' expectations.

We here distinguish between two kinds of contextual information. On the one hand there is what we refer to as raw contextual information. On the other hand, this contextual information will be the basis for derived or higher-level contextual information. Raw contextual information will come from two main sources: i) direct user interaction with applications and infrastructural services, and ii) monitoring logs. The first kind of contextual information refines and enhances the user profile by gathering details of the interaction during the creation and consumption of services. For example, imagine a user who plans a trip and uses a travel Web site which provides transportation and accommodation services. There will be several combinations that will allow a trip from a source location to a destination depending on the user preferences and profile. Any feedback on previous experiences, relationships of trust with services providers, based on previous trips will allow to customize and presumably enhance the user experience by providing services that are expected to be more appropriate [59]. Once the services are delivered, the user will be given the opportunity to provide feedback about his or her experience in order to obtain a better experience in future interactions.

The other highly valuable source of contextual information will concern services more specifically. This information will be provided by service providers, service consumers and the monitoring infrastructure and it will contribute to obtaining comprehensive service profiles. Services characterisation will be based on their descriptions, information concerning their execution and additional semantic annotations that users may attach to them. Services descriptions will inform us about the provider, the data manipulated, their intended purpose, etc. Previous executions will provide us additional valuable monitoring information concerning their performance, their reliability, their use within other composite services, etc. Finally, the Internet as a social platform will contribute to the overall body of knowledge concerning services by means of user tags, annotations, previous uses of the services by other consumers, etc.

On the basis of this raw contextual information, a large quantity of higher-level, derived, and presumably more useful information will be generated, see Figure 6. For instance using monitoring information we can derive metrics concerning their execution, e.g., "average execution time", the expected reliability. Using information tracing users behaviour we can derive statistical information in order to find out the user preferences. Using global information about users and services we can define clusters of similar users and similar services, etc. Having this derived quantitative and qualitative information concerning users and services, we can support presenting services to the user categorised by the task they perform (may this be explicitly stated or derived implicitly). Compatible services can be ranked and selected at runtime based on the perceived performance or reliability. We could support suggesting services based on profiles similarities between users or the perceived trust based on social networks, and service execution will be able to automatically adapt to contextual information such as for instance the kind of customer being served.

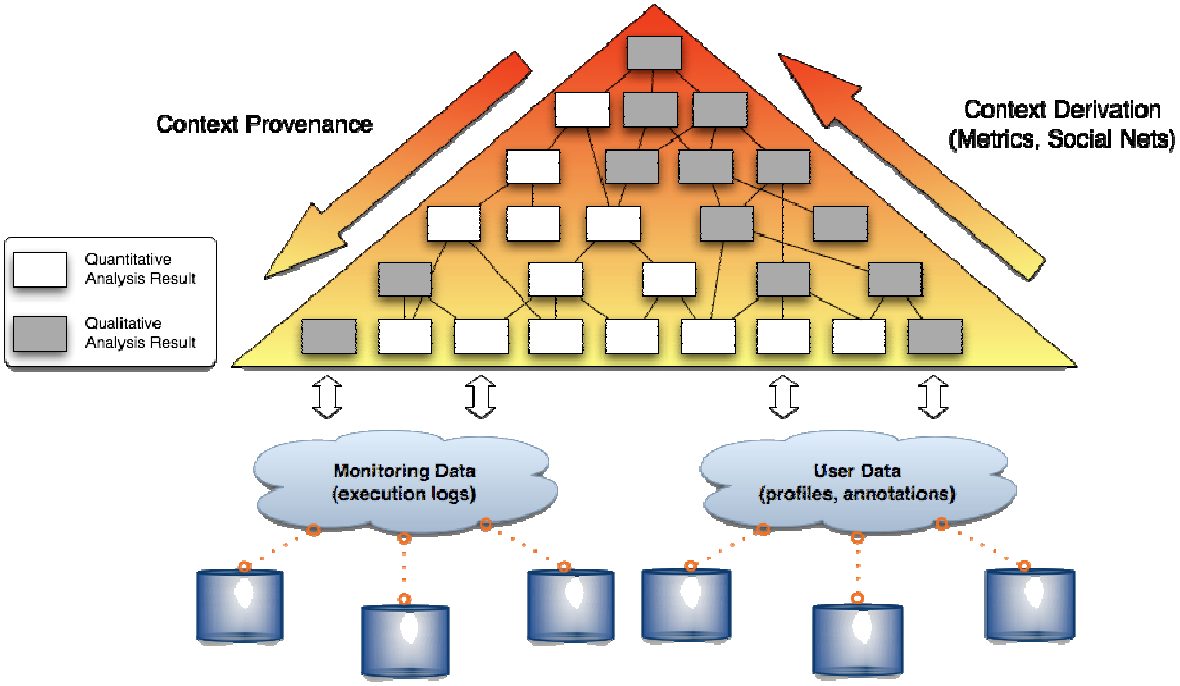

Figure 6. Contextual information derivation and provenance.

4.1.4 Context Tracing

One fundamental issue is to have some capability for tracing context, that is to have a clear understanding where execution context is from, and reason about it. While our Context Recognition Service is at runtime (see Section 4.2), we here focus on a post-mortem analysis in order to facilitate the understanding and feedback of complete process executions. However, this is a very complex task, because it means to deal with knowledge-intensive situations at different levels of detail: from technical ones (e.g. messages exchanged) to higher-level information regarding the service delivery (e.g. performance information, economical impact, etc).

This context tracing capability can be very helpful for two reasons: firstly, because we could use such knowledge as a feedback for next service adaptations and update of the user context accordingly; secondly, because we could use such knowledge to provide users with more meaningful interpretations of process executions context, in a way closer to how domain experts reason on a particular problem and facilitating their comprehension.

As a consequence, we approach the task of context tracing based on the use of provenance. The goal of provenance is to provide a better understanding of knowledge business processes that allows validating such processes and eventually improving performance and quality of service, eventually easing computation from the human perspective.

Provenance is broadly defined as the origin or source from which something comes, and the history of subsequent owners (also known in some fields as chain of custody). This term has been used traditionally in areas like archaeology and palaeontology, where this type of information is very relevant to help determine whether a piece of art or a finding is real or not.

The use of this term in other scientific and non-scientific disciplines has extended its scope to capture other aspects, such as place and time of manufacture, production or discovery; quality of the transformation processes if any, price, etc. And a plethora of comparative techniques, use of expert opinions, written and verbal records and results of tests are often

used to help establish provenance in all these different disciplines.

In the context of data, process and computation-intensive disciplines, such as physics, biology, astronomy, etc., to name but a few, provenance is focused on the description and understanding of where and how data is produced, the actors involved in the production of such data, and the processes applied to the object before arriving in the collection from which it is now being accessed, so that it can be considered as an important source of information to determine its overall quality. For instance, in a usual discovery task, scientists integrate data from data sources, filter the combined data according to some criteria, and annotate the data with information about the relationships that have just been discovered. All the tasks applied in this process contribute to the provenance record of that data product.

However, having all this information recorded together with the data product is not usually enough, since it requires an abstraction process before it can be actually used, given the large amount of information that can be recorded. According to [60], provenance information can be seen as a pyramid with four main levels: Data, Organization, Process, and, on top of the pyramid, Knowledge. While most of the current provenance systems are focused on the first three levels of this pyramid, providing means for recording and querying of process documentation, other efforts, e.g., myGrid [60], approach the provenance problem from the semantic perspective, tackling with the knowledge level of the aforementioned pyramid. These systems exploit semantic technologies to provide more expressive means to describe provenance by means of domain ontologies represented in Semantic Web knowledge representation languages, like RDF(S) and OWL, which establish well-defined associations between the resources used during process documentation and the domain. This allows building semantic provenance metamodels that provide the terminology necessary to meaningfully express provenance entities and the relations between them. In summary, knowledge-oriented provenance systems intend to facilitate user understanding and comprehension of process executions in a variety of domains, e.g. biology or business, by explaining provenance in the own terms of such domains. From now on, we will focus on this group of systems.

Independently of the approach taken for provenance gathering and representation, upon documenting a process execution, large quantities of highly linked and annotated provenance data will be generated. When the size and complexity of the processes increase, process documentation can become hard to assimilate and eventually unmanageable. Therefore, there is an issue about whether both the presentation and computational handling of these data are scalable. Furthermore, the main beneficiaries of provenance information are domain experts (e.g. biologists, telecommunications experts, etc.), who do not necessarily have a strong background in computer science and, more specifically, provenance. Thus, an additional semantic layer with a higher level of abstraction can be brought in that helps leveraging this problem.

In the SOA4All case, there are two main complementary places where the usage of Provenance can help to recognise the context at hand in the Context Recognition Service within Heuristic Classification. Figure 7 depicts how Provenance can help Context Recognition Service to update user contexts.

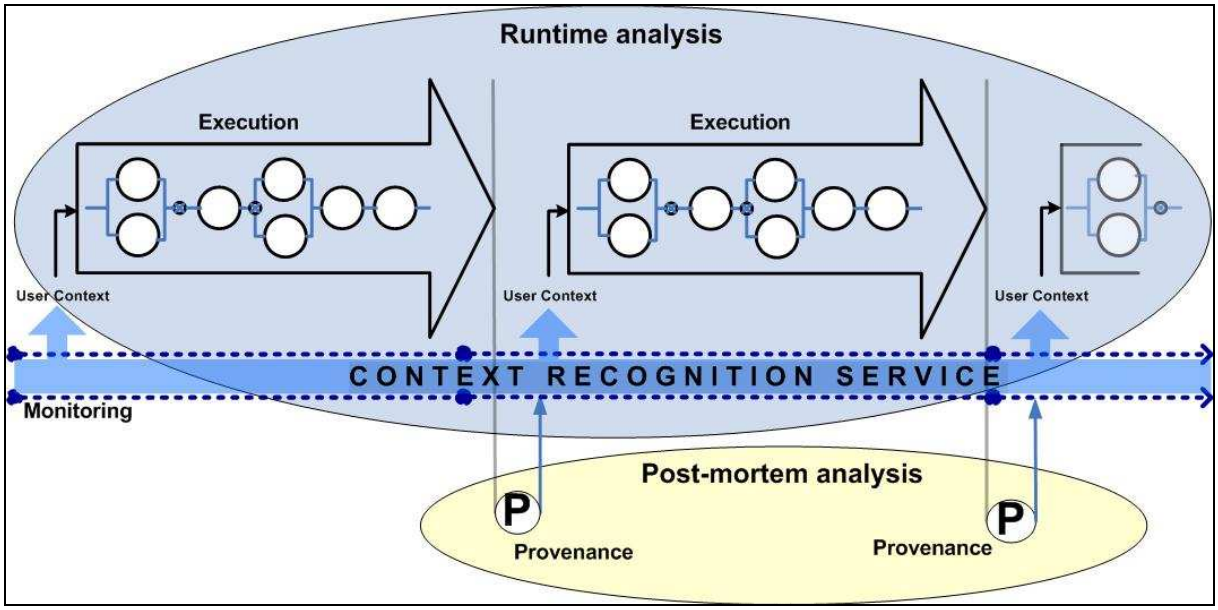

Figure 7. Relationship between Knowledge Provenance and Context Recognition.

In concrete, there are two complementary services that Provenance can provide to help to the Context Recognition.

On the one hand, a Data Provenance Service can trace the origin and nature of the concepts used by context derivation techniques (metrics computation, social networks-based results, etc) in order to identify the origin of the data and thus compute some kind of quality and trust. To ensure that data retrieved from different sources is used appropriately and within context, it is imperative that the provenance of the data be recorded and made available to the service [68]. This information is typically lost in the process of transforming and reasoning, and such trust measure can be used to state the uncertainty over the sources of the data gathered or abstracted by the Context Recognition Service.

On the other hand, a Knowledge Provenance Service using KOPE (see D2.3.1) can facilitate the understanding of the complete process execution, offering an overall decomposition into domain-level subprocesses, and providing a new compliant trust measure over a set of process templates. As well as we intend to use the measure provided by the Data Provenance Service, the measure provided by the Knowledge Provenance Service will be used as a criterion in the Context Recognition Service.

4.2 Context Recognition Service

We understand Context Recognition as a knowledge intensive task that, given a body of information capturing contextual information about services and users, and a particular purpose for which a context has to be identified, it returns the concrete context recognised. For example, given a user profile, determining to which category of customers the user belongs to (e.g., highly-valued customer, normal customer, new customer), is a matter of generating a classification problem specification that is solved by a general purpose Classification PSM. The result obtained from the Classification PSM, i.e., the kind of customer we are actually dealing with, provides us the required contextual knowledge (e.g., highly-valued customer) for adapting our service (e.g., lower prices, faster delivery, etc). This process is illustrated in Figure 8.

We therefore approach the task of determining a (set of) relevant context(s) as a process involving capturing contextual information, deriving additional information, and classifying it

with respect to general cases. In fact, it is this very process of classifying contextual information that produces *contextual knowledge* which can then support the appropriate adaptation of services. The reader should notice that we are here distinguishing between *contextual information* (i.e., all the contextual information gathered may it be useful in the situation at hand or not) and *contextual knowledge* which can support the system acting rationally [46]. In this process we also contemplate lifting low-level syntactic data into a semantic format that can support further reasoning as necessary for applying the PSM.

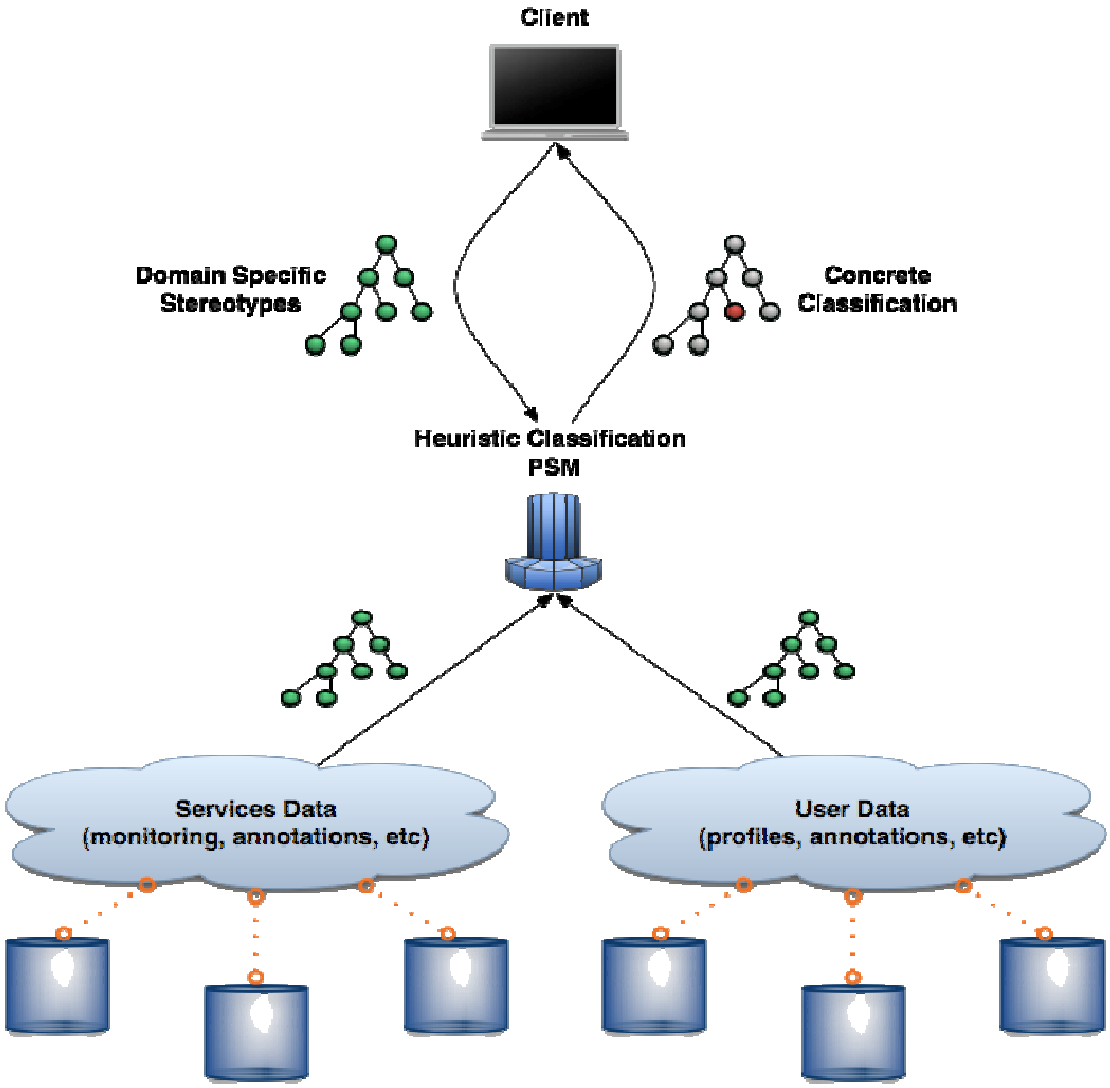

Figure 8. Context Recognition as Heuristic Classification.

4.2.1 Heuristic Classification

Classification is a knowledge-intensive task that we encounter and perform (often unconsciously) in many of our daily tasks. In general terms, classifying is "to understand something as a specific instance of a more general case—which is what understanding a more fundamental principle or structure means—is to have learned not only a specific thing but also a model for understanding other things like it that one may encounter" [61].

Heuristic Classification is a particular technique for classifying that was identified, thoroughly analysed and extensively described by Clancey in his seminal paper [44]. Heuristic Classification is a Knowledge Level [46] method for classifying entities that "*relates data to a pre-enumerated set of solutions by abstraction, heuristic association, and refinement*" [44] (see Figure 2). For instance diagnosing diseases has often been approached in Artificial Intelligence as a process whereby basic observations about patients like the temperature (e.g., 39º C) are abstracted (e.g., temperature is high), matched against prototypical diseases (e.g., the flu), and refined so that the concrete diagnosis is the one that best explains the symptoms observed. Indeed, the previous scenario has been kept simple for clarity purposes. Realistic scenarios include a wide range of observations (e.g., fever, vomiting), which may have causal relations (e.g., high temperature might cause a certain disorientation), and all need to be reconciled to obtain a correct classification (e.g., fever and vomiting because the patient has gastroenteritis).

In most of the situations, the solution features will not be directly provided by the data observed. Instead they need to be abstracted from the raw data that is available. There are three different kinds of abstraction:

- *Definitional abstraction* is based on the essential characteristics of the concept being analysed (e.g., a computer scientist knows about computers).
- Qualitative abstraction is the derivation of some qualitative feature based on quantitative measures (e.g., if service A takes longer than 20 seconds it is performing slowly)
- Generalization is based on the use of hierarchical information (e.g., a man is a mammal)

Refinement is a similar process but the direction of the inference is inverted. The goal is to, once a prototypical class has been identified, refine the solution trying to explain most of the observed facts. It is worth noting that refinement takes place within the target classification ontology trying to explain all the fact observed.

The most distinctive feature of Heuristic Classification is the fact that a heuristic is used for achieving a non-hierarchical direct association between the abstracted data and the prototypical classes contemplated. The knowledge for relating heuristically problem situations to solutions that tend to work is essentially experiential. A heuristic relation is uncertain, based on certain assumptions of commonality, and allows a reduction in search by skipping over intermediate relations. Heuristic Classification does not attempt to generate a solution that has not previously been identified and therefore, should the prototypical classes be too vague or the scope of the domain they capture too narrow, the solution obtained will also be affected.

4.2.2 Heuristic Classification Problem Solving Method

Previous research in Knowledge Engineering has been devoted to the formalisation of Heuristic Classification as a reusable domain-independent Problem Solving Method. In particular, in [62], the authors present a library of components for classification problem-solving. Their work includes a formalisation of Classification as a knowledge-intensive task and a set of prototype components for solving Classification problems. The library is based on the Task Method Domain Application (TMDA) framework [40]. TMDA prescribes constructing Knowledge-Based Systems based on the definition of task ontologies that define classes of applications (e.g., diagnosis, classification), method ontologies that capture the knowledge requirements for specific methods (e.g., heuristic classification), domain ontologies that provide reusable task-independent models, and application ontologies for integrating domain models with domain-independent problem solvers. Hence, the library defined in [62] includes a task ontology and a method ontology. Domain and application

ontologies should be defined on a per-application basis.

We shall therefore build upon the existing work for supporting Context Recognition within our Service Adaptation Framework. The devised ontology stack for modelling contextual information together with concrete context information provided by users or automatically obtained by the context derivation facilities will constitute the domain ontology. As part of our service adaptation framework we will include an application ontology that will relate our context domain ontologies to the classification ontologies in order to be able to solve classification problems.

Classification task ontology characterises the task as the problem of finding the solution (a concept) which best explains a certain set of facts (observables) about some individual, according to some criterion. *Observables* are basically facts which are known about the individual we want to classify. Observables are defined as pairs of the form <feature, value>. The *solution space* specifies a set of classes under which the individual to classify may fall. Each *solution* is itself specified as a set of pairs of the form <feature, condition>, whereby the condition determines the values the feature can take. An observable is said to match a feature specification if the value satisfies the condition. Based on these definitions there are different solution criteria that one can adopt. We may accept solutions that explain some of the data and have no inconsistent features (the value doesn't satisfy the condition). We may require a complete coverage, which is the case when all the observables are explained consistently. In some other cases we may be looking for the best solution rather than admissible ones. Ranking the solutions relies on what we refer to as a *match criterion* which is basically a solution scoring mechanism. The specification of a general purpose Classification Task is depicted in Figure 9.

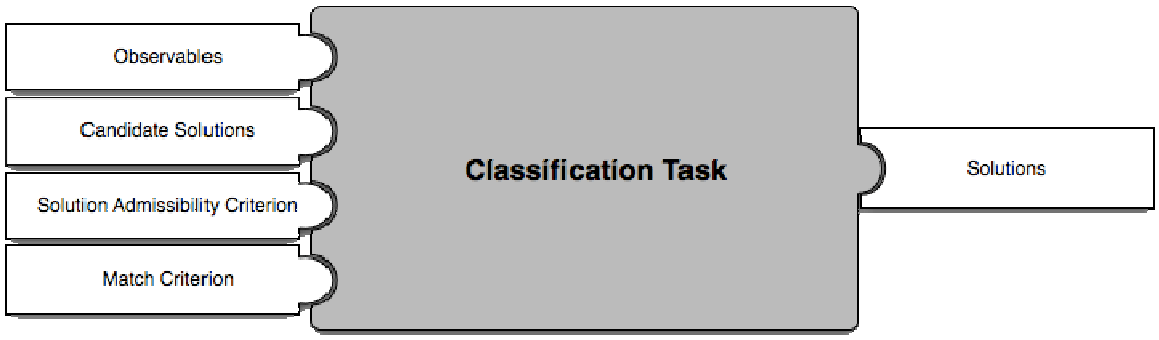

Figure 9. Classification Task.

In addition to the generic task definition, [62] includes a set of Problem-Solving Methods for classification based on the Heuristic Classification model previously presented. The library includes a Problem-Solving Method for obtaining a single admissible solution and another one for obtaining an optimal solution. The former extends the inputs of the Classification Task with a set of abstractors, refiners and a *solution exclusion criterion* which allows pruning the search space when certain exclusion criteria are identified. The library approaches classification as a data-driven search where observables try to be explained in accordance to some given criteria. Additional observables can be derived by abstraction but no raw observables can be incrementally acquired. Abstraction is supported by so-called *abstractors* which are basically functions that given a set of observables produce additional observables. Conversely, refinement is supported by so-called *refiners* which can be seen as the inverse of abstractors. Refiners themselves are also defined as functions that given the solution space refine it towards the solution. Coming back to our previous discussion concerning context derivation, it is we quite clear that the context derivation techniques (metrics computation, social network-based results, etc) are abstractors within our framework.

So far we have based our explanation on a typical scenario where this method is applied for medical diagnosis. However, this very method has been used for a wide variety of purposes such as device diagnosis, catalogue selection, model-based analysis, or even user profiling [44]. This last use case is particularly relevant for our purposes since, one of the kinds of contextual information we shall manage concerns precisely users. Indeed, being able to appropriately find stereotypes of users is crucial for achieving the kinds of service adaptation we aim at within SOA4All.

4.3 Service Adaptation

Once a relevant context or sets of contexts have been identified, services need to be adapted accordingly. The last infrastructural service provided by the Contextual Service Adaptation framework is precisely in charge of this. The main drivers for requesting the adaptability of services are due to the scale of the vision of SOA4All. Creating a Web of billions of services will give rise to an outstanding heterogeneity at the level of services. Without proper mechanisms for abstracting this heterogeneity away, the management of the services available will impede on the one hand a proper maintenance and evolution of services, and on the other hand it will difficult the task of finding interesting or suitable services for a given purpose.

SOA4All uses as a basis WSMO and derived light-weight approaches that will be defined within the project. WSMO provides the means for defining Semantic Web Services by providing four main building blocks, namely Ontologies, Goals, Web Services, and Mediators. Of those four building blocks Web Services and Goals are the main abstractions that are relevant for the adaptation of services. The former represent semantic models of traditional web services with formalised interfaces and capabilities. The latter can be seen both as an abstraction over user's goals, hence the name, and as an abstraction over a set of services.

Like in real life, Goals can most often be achieved in several ways. The "best" one and even those that are suitable depend on a variety of aspects, many of which are merely contextual. Imagine for instance that we want to pay a certain object in an Internet shop. This could be achieved using different kinds of credit cards or even systems like Paypal. The suitability will depend on the kind of cards we have available, on whether we have a Paypal account, etc. The "best" solution for us will depend on our preferences in terms of cost, security, simplicity, etc. These factors are all contextual since the essence of the service, that is to pay a certain amount of money, remains unchanged

Goals provide us therefore with the required level of abstraction able to introduce contextual aspects within the execution of services. In simple cases, like the one outlined above, adapting services to a certain context will be a matter of introducing within the execution infrastructure [15, 63] the capability to recognise a relevant context (see Section 4.2) and use this information to restrict the suitable services and select the best option. In more complex cases like composite Goals (those composed of several sub goals), contextual factors of one Goal may have implications over others. Imagine for instance that we want to contract an Internet connection and a VoIP solution with a certain Quality of Service (QoS). The kind of connection chosen might not support the QoS desired, yet an Internet connection must be in place before contracting the VoIP solution. Supporting this simple process in a completely automated way reveals to be particularly complex since context changes provoked by the first Goal (i.e., get an Internet connection) might prevent us from achieving the second Goal (i.e., get a VoIP with a certain QoS). Indeed, it is quite easy to envisage situations where the interdependencies are such that the complexity of the problem cannot be neglected.

In order to cater for this kind of situations and still maintain the overall objective of the project, that is to provide general purpose solutions able to scale to a Web of billions of services, we envisage approaching this problem as a parametric design problem. This

approach is similar to that of [47] although we aim to provide a general purpose infrastructural solution that will allow Semantic Execution Environments like the IRS-III or WSMX [15, 63] to execute contextualised Semantic Web Services in a generic way. In the reminder of this section we shall present an overview about parametric design and more concrete details about a parametric design PSM.

4.3.1 Parametric Design

Designing is one of the most remarkable intelligent behaviours exhibited by humans. In Smither's words [64] "what makes designing a particular kind of activity, distinct from problem solving and planning, and other human activities, is that designing must start with something that neither specifies what is required nor defines a problem to be solved, yet it must arrive at a design specification for something that, when realised or implemented, will satisfy the motivating needs or desires: the realised design should remove the need or desire for something to be different."

It is a particularly interesting task for its high complexity and the usual lack of any established methodology to guide designers (i.e., a human or a computer-based system) through the whole process towards a suitable result. Since Design is a very ample task type, many other kinds of problems can be approached as a design task. However, there are obvious inefficiencies that would arise in the cases where, the full power of designing is not required. Perhaps the prototypical example is Configuration Design where many task-specific approaches have been proposed in order to increase systems performance. Configuration Design is often defined as [65] "*... a form of Design where a set of pre-defined components are given and an assembly of selected components is sought that satisfies a set of requirements and obeys a set of constraints.*" Typical examples of Configuration Design tasks can be configuring a computer, a car or an elevator (which is perhaps the most explored domain in Configuration Design literature) [65].

The general case of Configuration Design assumes that components can be freely arranged which in many cases is too strong an assumption. In fact, designers often have a predefined idea of what a solution is going to look like. For example, when configuring a computer, experts know that there will be a processor, a motherboard, a hard drive, etc. In many other cases it is even possible to have a parametrised solution template. In these cases, design problem-solving consists of assigning values to the parameters taking into account a set of needs, constraints, and desires. This kind of problems, which we refer to as Parametric Design problems, greatly decrease the complexity of the problem-solving task with respect to other types of designing [40].

Much research has been devoted to design problem-solving in Artificial Intelligence, especially for the Configuration and Parametric cases [40, 45, 65-68]. Some researchers have focussed on theoretical characterisations of designing whereas others have focussed on the development of systems that can support it. Among the latter, one of the most comprehensive toolsets [40] was formalised and developed on top of the TMDA framework previously introduced. It therefore includes a task ontology that characterises parametric design, and a method ontology that includes a set of methods for solving parametric design problems.

Therein the parametric design task is defined as a task that given a set of *parameters*, a set of *constraints*, a set of *requirements*, a *cost function*, a *cost algebra*, and a set of *preferences*, obtains a *design model,* see Figure 10. Design models, that is the solutions to a parametric design problem, are defined as a set of parameters assignments of the form <parameter, value>. Constraints specify conditions that need to be satisfied by a design, and therefore restrict the solution space. Requirements on the other hand describe desired properties solutions should have. Requirements are therefore the positive counterpart of constraints. Preferences are basically the means for ranking the different solutions obtained. They are therefore different from requirements and constraints since they do not restrict the

space of possible solution but rather establish an order between the solutions obtained. Finally, the cost function and the cost algebra simply support the introduction of a global preference system for ordering solutions based on the combination of all the preferences provided.

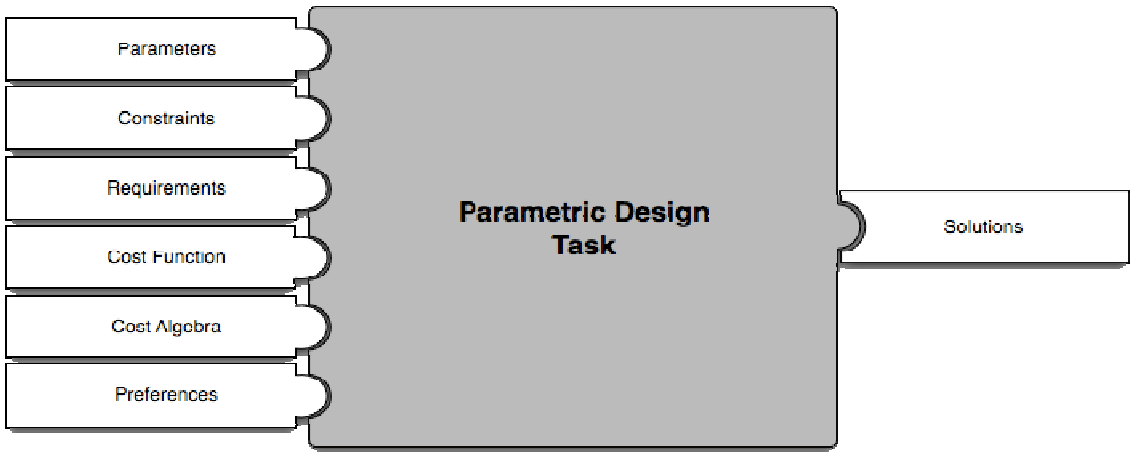

Figure 10. Parametric Design Task.

In addition to the generic task definition, [62] includes a set of Problem-Solving Methods for parametric design. The library includes a general purpose search-based Problem-Solving Method based on the notions of *design state* and *design operators*. In this case parametric design is approached as the exploration of the possible design states which can be reached by applying the existing design operators. More advanced problem-solving strategies such as *Propose & Backtrack* or *Propose & Revise* are also investigated in order to achieve further performance by including additional knowledge for guiding the exploration of the solution space. The trade-off indeed lays on the fact that more advanced problem-solving techniques require additional knowledge and the knowledge acquisition process constitutes a considerable overhead.

In this respect, future work within the work package will be devoted to i) integrating the parametric design conceptualisation with the overall context ontology stack; ii) implementing the Service Adaptation service on top of a Parametric Design PSM; and iii) investigating the adaptation of service execution frameworks to make use of the Service Adaptation service at runtime.

5. Conclusion

SOA4All intends to enable a world where billions of users are exposing and consuming services. Supporting the adaptation of services based on contextual information plays a crucial role to achieve this vision. Context is herein understood as *any information that can be used to characterize the situation of an entity and better support the interactions between services and between users and services.*

The approach we follow to context adaptation is based on a set of conceptualisations providing the means for modelling contextual information of any kind, and a general purpose machinery able to manage contextual data, use it for recognising concrete contexts within particular situations, and apply this contextual knowledge for adapting services. Therefore, central to our approach, much like the definition of context itself, is the genericity of our framework so that it can cover the wide range of situations one is likely to encounter in a Web of billions of services.

The framework will provide three main generic services to the overall SOA4All vision: Context Management, Context Recognition and Service Adaptation based on Context. To do so, we shall base our development on the use of PSMs for capturing the dynamic knowledge for supporting Context Recognition and Service Adaptation in a domain-independent way, and ontologies both for defining the interfaces of these services and for capturing contextual information in a sharable and machine processable way.

Context Management will be supported in our framework by means of an ontology stack that will provide the basis for capturing formally contextual data. Contextual information will be gathered both at runtime through monitoring and at design time based on user interactions. This context information will be managed by a federation of Context Repositories forming a cloud of distributed data, which will profit from the global scale of the infrastructure, to derive further information from emerging social networks in a Web 2.0 fashion. In fact, we foresee that this constellation of Context Repositories will give birth to new business models centred on the strategic business value of contextual information gathered over time by diverse organisations within particular communities.

Our epistemological basis for context information generation and context recognition is based on Heuristic Classification [44], a well-known Problem Solving Method [38] for solving Classification tasks [45]. Within our framework we approach the task of determining a (set of) relevant context(s) based on a wide range of information concerning users and services, as a process involving the abstraction of information gathered, its heuristic classification with respect to general cases, and the refinement to the particular case at hand. In fact, it is this very process of heuristically classifying contextual information that produces contextual knowledge which can then support the appropriate adaptation of services. The reader should notice that we are here distinguishing between *contextual information* (i.e., all the contextual information gathered may it be useful in the situation at hand or not) and *contextual knowledge* which can support the system acting rationally [46].

Once the particular context or sets of contexts have been determined, our framework will support the adaptation of services at design time, while creating composite services, and at runtime based on available contextual information at that particular moment. Again, we build upon the use of a generic Problem Solving Method to support this task. In particular, we will use Parametric Design [40]. In a similar vein to that presented in [47], we view the adaptation of services to particular contexts as a Parametric Design task where the parameters are context dependent features of the service such as the currency, the kind of user, etc.

These services will be offered by the Contextual Service Adaptation Framework as core infrastructure services for the SOA4All platform so that any other component, such as the Service Provisioning platform, can benefit from their functionality in order to support context adaptation.

6. References

1. Hohpe, G. and B. Woolf, *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*. 2003, Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.
2. Papazoglou, M.P., et al., *Service-Oriented Computing: State of the Art and Research Challenges.* Computer, 2007. 40(11): p. 38--45.
3. Hagen, C. and G. Alonso, *Exception Handling in Workflow Management Systems.* IEEE Transactions on Software Engineering, 2000. 26(10): p. 943-958.
4. Verma, K. and A. Sheth. *Autonomic Web Processes*. in *Service-Oriented Computing - ICSOC 2005*. 2005.
5. Menascé, D.A., *Composing Web Services: A QoS View.* IEEE Internet Computing, 2004. 8(6): p. 88--90.
6. Menasce, D.A., *QoS Issues in Web Services.* IEEE Internet Computing, 2002. 06(6): p. 72--75.
7. Winograd, T., *Architectures for Context.* Human-Computer Interaction, 2001. 16(2): p. 401 - 419.
8. Maamar, Z., D. Benslimane, and N.C. Narendra, *What can context do for web services?* Communications of the ACM, 2006. 49(12): p. 98--103.
9. Maamar, Z., et al. *Context-based Personalization of Web Services Composition and Provisioning*. in *30th EUROMICRO Conference (EUROMICRO'04)*. 2004. Los Alamitos, CA, USA: IEEE Computer Society.
10. Keidl, M. and A. Kemper. *Towards context-aware adaptable web services*. in *WWW '04: Proceedings of the 13th international World Wide Web conference on Alternate track papers \& posters*. 2004. New York, NY, USA: ACM.
11. Verma, K., et al., *METEOR-S WSDI: A Scalable P2P Infrastructure of Registries for Semantic Publication and Discovery of Web Services.* International Journal of Information Technologies and Management, 2005. 6(1): p. 17--39.
12. Sirin, E., B. Parsia, and J. Hendler, *Filtering and Selecting Semantic Web Services with Interactive Composition Techniques.* IEEE Intelligent Systems, 2004. 19(4): p. 42--49.
13. McIlraith, S., T. Son, and H. Zeng, *Semantic Web services.* IEEE Intelligent Systems, 2001. 16(2): p. 46--53.
14. Fensel, D., et al., *Enabling Semantic Web Services: The Web Service Modeling Ontology*. 2007: Springer.
15. Fensel, D., M. Kerrigan, and M. Zaremba, eds. *Implementing Semantic Web Services: The SESA Framework*. 2008, Springer.
16. Domingue, J., et al., *IRS-III: A broker-based approach to semantic Web services.* Web Semantics: Science, Services and Agents on the World Wide Web, 2008. 6(2): p. 109--132.
17. Galizia, S., A. Gugliotta, and J. Domingue. *A Trust Based Methodology for Web Service Selection*. in *International Conference on Semantic Computing, ICSC 2007*. 2007.
18. Dey, A.K., *Understanding and Using Context.* Personal Ubiquitous Computing, 2001.

5(1): p. 4--7.

19. Engelmore, R.S. and A.J. Morgan, *Blackboard Systems*. The Insight Series in Aritificial Intelligence, ed. R.S. Engelmore and A.J. Morgan. 1988: Addison-Wesley.
20. Brooks, R.A., *Intelligence Without Representation.* Artificial Intelligence, 1991. 47: p. 139--159.
21. Korpipaa, P., et al., *Managing Context Information in Mobile Devices.* IEEE Pervasive Computing, 2003. 02(3): p. 42-51.
22. Dey, A.K., G.D. Abowd, and D. Salber, *A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications.* Human-Computer Interaction, 2001. 16(2): p. 97 - 166.
23. Hong, J.I. and J.A. Landay, *An Infrastructure Approach to Context-Aware Computing.* Human-Computer Interaction, 2001. 16(2): p. 287--303.
24. Strang, T. and C. Linnhoff-Popien. *A Context Modeling Survey*. in *First International Workshop on Advanced Context Modelling*. 2004. Nottingham, UK.
25. Krummenacher, R., H. Lausen, and T. Strang. *Analyzing the Modeling of Context with Ontologies*. in *International Workshop on Context-Awareness for Self-Managing Systems*. 2007.
26. Guarino, N. and C. Welty, *Evaluating ontological decisions with OntoClean.* Communications of the ACM, 2002. 45(2): p. 61--65.
27. Gómez-Pérez, A., *Evaluation of ontologies.* International Journal of Intelligent Systems, 2001. 16(3): p. 391--409.
28. Khedr, M. and A. Karmouch, *Negotiating Context Information in Context-Aware Systems.* IEEE Intelligent Systems, 2004. 19(6): p. 21--29.
29. Strang, T., C. Linnhoff-Popien, and K. Frank, *CoOL: A Context Ontology Language to Enable Contextual Interoperability*, in *Distributed Applications and Interoperable Systems*. 2003. p. 236--247.
30. Beaune, P., O. Boissier, and O. Bucur. *Representing Context in an Agent Architecture for Context-Based Decision Making*. in *Context Representation and Reasoning Satellite Workshop of CONTEXT'05*. 2005.
31. Heckmann, D., et al. *Gumo – The General User Model Ontology*. in *10th International Conference on User Modeling*. 2005.
32. Chen, H., et al. *SOUPA: Standard Ontology for Ubiquitous and Pervasive Applications*. in *International Conference on Mobile and Ubiquitous Systems: Networking and Services*. 2004. Boston, MA.
33. Wang, X.H., et al. *Ontology based context modeling and reasoning using OWL*. in *Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops*. 2004.
34. Heckmann, D., et al. *Gumo – The General User Model Ontology*. in *10th International Conference on User Modeling*. 2005.
35. Strang, T., C. Linnhoff-Popien, and K. Frank, *CoOL: A Context Ontology Language to Enable Contextual Interoperability*, in *Distributed Applications and Interoperable Systems*. 2003. p. 236--247.
36. Lum, W.Y. and F.C.M. Lau, *A Context-Aware Decision Engine for Content Adaptation.* IEEE Pervasive Computing, 2002. 01(3): p. 41--49.

37. Roman, M., et al., *A Middleware Infrastructure for Active Spaces.* IEEE Pervasive Computing, 2002. 01(4): p. 74--83.

38. Studer, R., R. Benjamins, and D. Fensel, *Knowledge Engineering: Principles and Methods.* Data Knowledge Engineering, 1998. 25(1-2): p. 161-197.

39. Staab, S. and R. Studer, eds. *Handbook on Ontologies*. International Handbooks on Information Systems. 2004, Springer.

40. Motta, E., *Reusable Components for Knowledge Modelling. Case Studies in Parametric Design Problem Solving*. Frontiers in Artificial Intelligence and Applications. Vol. 53. 1999: IOS Press.

41. Fensel, D., et al., *The unified problem-solving method development language UPML.* Knowledge and Information Systems, 2003. 5(1): p. 83--131.

42. Crubezy, M., et al., *Configuring Online Problem-Solving Resources with the Internet Reasoning Service.* IEEE Intelligent Systems, 2003. 18(2): p. 34--42.

43. Benjamins, R., *Problem-Solving Methods for Diagnosis and their Role in Knowledge Acquisition.* International Journal of Expert Systems: Research \& Applications, 1995. 8(2): p. 93--120.

44. Clancey, W.J., *Heuristic classification.* Artificial Intelligence, 1985. 27(3): p. 289--350.

45. Schreiber, G., et al., *Knowledge Engineering and Management: The CommonKADS Methodology*, ed. G. Schreiber. 1999: MIT Press.

46. Newell, A., *The Knowledge Level.* Artificial Intelligence, 1982. 18(1): p. 87--127.

47. ten Teije, A., F. van Harmelen, and B.J. Wielinga. *Configuration of Web Services as Parametric Design*. in *EKAW*. 2004. Whittlebury Hall, UK: Springer.

48. Sun, J.-Z. and J. Sauvola. *Towards a conceptual model for context-aware adaptive services*. in *Proceedings of the Fourth International Conference on Parallel and Distributed Computing, Applications and Technologies, PDCAT'2003.* 2003.

49. Shkundina, R. and S. Schwarz. *A Similarity Measure for Task Contexts*. in *Proceedings of the International Conference on Case-Based Reasoning (ICCBR'05)*. 2005.

50. Brickley, D. and L. Miller, *FOAF Vocabulary Specification 0.91*. 2007, http://xmlns.com/foaf/spec/.

51. Colombo, E., J. Mylopoulos, and P. Spoletini. *Modeling and Analyzing Context-Aware Composition of Services*. in *Service-Oriented Computing - ICSOC 2005*. 2005.

52. Sutcliffe, A., S. Fickas, and M.M. Sohlberg, *PC-RE: a method for personal and contextual requirements engineering with some experience.* Requirements Engineering Journal, 2006. 11(3): p. 157--173.

53. Gruber, T.R., *A translation approach to portable ontology specifications.* Knowledge Acquisition, 1993. 5(2): p. 199--220.

54. Musen, M.A., *Ontologies: Necessary--Indeed Essential--but Not Sufficient.* IEEE Intelligent Systems, 2004. 19(1): p. 77--79.

55. McBride, B., *The Resource Description Framework (RDF) and its Vocabulary Description Language RDFS*, in *Handbook on Ontologies*, S. Staab and R. Studer, Editors. 2004, Springer-Verlag. p. 51--66.

56. Dean, M., et al., *OWL Web Ontology Language Reference*. 2004, http://www.w3.org/TR/owl-ref/.

57. Motta, E. and W. Lu, *A Library of Components for Classification Problem Solving*. 2001, The Open University.

58. OASIS, *OASIS Web Service Context Specification*. 2007, http://docs.oasis-open.org/ws-caf/ws-context/v1.0/wsctx.html.

59. Lin, N., Kuter U. and E. Sirin. *Web Service Composition with User Preferences, ESWC 2008.* in *Proceedings of the European Semantic Web Conference*. 2008.

60. Zhao, J., et al. *Using Semantic Web Technologies for Representing e-Science Provenance*. in *Procedings of the 3rd Internacional Semantic Web Conference (ISWC2004)*. 2004.

61. Bruner, J.S., *The Process of Education*. 2002, Cambridge, Massachusetts: Harvard University Press.

62. Motta, E. and W. Lu, *A Library of Components for Classification Problem Solving*. 2001, The Open University.

63. Norton, B., et al., *Semantic Execution Environments for Semantics-Enabled SOA.* it - Methods and Applications of Informatics and Information Technology, 2008. Special Issue in Service-Oriented Architectures: p. 118--121.

64. Smithers, T. *On Knowledge Level Theories and the Knowledge Management of Designing*. in *International Design Conference - Design (2002)*. 2002. Dubrovnik.

65. Schreiber, A.T. and B.J. Wielinga, *Configuration Design Problem Solving.* IEEE Expert, 1997. 12(2): p. 49--56.

66. Chandrasekaran, B., *Design problem solving: a task analysis.* AI Magazine, 1990. 11(4): p. 59--71.

67. Chandrasekaran, B., *Generic tasks in knowledge based reasoning: high-level building blocks for expert system design*, in *IEEE Expert, Fall 1986*. 1986. p. 23--30.

68. Pedrinaci, C., *Knowledge-Based Reasoning over the Web. Phd Thesis.* 2005, Universidad del País Vasco/Euskal Herriko Unibertsitatea.