

Chapter 2

The Role of Semantics in ICT at the Cloud Era

2.1 Introduction

This chapter introduces overall foundations to identify the role of knowledge engineering area and particularly semantic aggregation activity in the ICT and communication areas and their influence for a possible convergence to support communication services. The described foundations guide you for constructing a conceptual framework around a particular problem in managing communication services which is endowed to the information used for management purposes. The semantic expressiveness is necessary for supporting Internet services, allows the customization, definition, deployment, execution and maintenance of services, and then improves the management functions and operations in multiple networks and service applications. The role of semantic applications which support these services can be used in ICT and telecommunications services where context-aware and autonomic features are also discussed.

Particularly, this chapter defines the basic concepts and requirements that pervasive service applications impose as requirements on network and services infrastructures from a management point of view. This chapter makes references to a wide range of context integration research, followed by an analysis of policy-based management (PBM) models [[IST-CONTEXT](#)] and cloud services and infrastructures.

The organization of this chapter is as follows. Section [2.1](#) presents the importance of using ontologies in pervasive services and management operations, in a way of initial description for understanding the importance of semantics in the cloud era.

Section [2.2](#) introduces fundamental concepts and terminology used. The definitions of terms related to diverse domains, such as context-awareness and context modelling, service management and operations, ontology engineering and autonomic communications, virtual infrastructures and cloud computing, with the objective of building the conceptual platform are used in this book.

Section [2.3](#) addresses the need to understand and recognize ontology, the importance of providing semantic-oriented and pervasive solutions, and highlights why

the integration of context information, with formal semantic mechanisms as ontologies, in service operations to support network and service management, is relevant for cloud services management. At the same time and briefly this section is intended to present trends in pervasive management to understand how semantics are being used in autonomic communications and cloud computing.

Section 2.4 discusses the semantic aspects of knowledge engineering and context information in the form of ontology operations and communications services supporting cloud computing. Challenges regarding context information, information models, service management and ontologies are discussed in terms of what, why, where and most importantly, how those concepts are part of the cloud computing challenges.

This section acts as the state of the art for the requirements in current IT infrastructures and cloud service management. This section introduces the major areas related to information requirements, end-user requirements and technological requirements, and discusses cloud computing challenges and trends. Commonalities are highlighted to produce a compilation or research challenges list and technological requirements.

Section 2.5 introduces the chapter conclusions in form of a discussion about concepts and ideas from the diverse challenges and trends in cloud computing introduced in this chapter.

2.1.1 Importance of Using Ontologies in Pervasive Services and Management Operations

In communication systems, pervasive computing (frequently named ubiquitous computing indistinctly), plays an important role when information processing and management is required. First because it has an inherent feature and/or ability to take advantage of collected pieces of information and related them with service applications and second because in pervasive computing the context-awareness is considered as the main aspect at the beginning of the system design. The advantages that context-awareness brings are principally in the automation of operations (e.g. in the support of activities related to self-management operations in communication networks), when new context information can trigger the deployment of new services improving their performance or developing new services or applications for end users.

It is clear that the complexity for designing and deploying support systems (i.e. customization, management and billing) for context-aware services is also high and requires appropriate tools and adequate infrastructures. One of the most difficult aspects in managing context-aware services is gathering the context information. There are many different proposed standard formats of context information, including people profile, location, network properties or status, applications and other application-specific information as example. This poses the challenge of tracking different mechanisms to gather the information and process it in real time, and then react according to those changes. Another and more important challenge is the

modelling and structure of the context information. Without a model, applications will not be able to use such information. Nevertheless, this model must be rich and flexible enough to accommodate not only the current but also future aspects of context information. Another important aspect to consider is that the model should be based on standards as much as possible and it should scale well with the network or the application.

The clear challenge is how modelling and structuring of the context information to gather, distribute and store the data. Context information in the framework of this book refers to operations in management functions. Within the vision of this book, with information models to represent and for integration of management operations and pervasive applications, ontologies play the role of those formal mechanisms that are able to model and use such information in a way that management operations come up as a result of more flexible and efficient creation, adaptation and deployment of communications services. Ontologies are essentially the tool to enhance, semantically, pieces of information and make them accessible to multiple and diverse applications.

The ontology-based model representing context information model must be self-describing, flexible and formal enough for representing not only the current status of the managed object, but also the future aspects that could be defined later as context information. The model should scale well with the network or the application domain. Most current applications use data models (and sometimes, though rarely, information models) for defining context information. An information model is usually not used, since these models are adapted for specific applications, and do not provide enough descriptive and semantic enrichment to model the interaction between applications at different abstraction levels to support cross-layered, interoperable operation.

2.2 Important Concepts and Terminology

The set of definitions, as they are being used in this book, are presented in this section.

2.2.1 *Context Information*

As an introductory premise, it is necessary to know that context information has been under research for long time [Dey97]. One of the most popular definitions of context introduces the concept entity and the interactions between them [Dey00b]. Due to its dynamicity and huge variety, context information cannot be defined as static information, so then it is not difficult to imagine the multiplicity of definitions aiming to capture part of the context features. Normally context definitions are associated with the application where the information is used and processed, rather than with the own context concept [Bauer03], [Brown97]. In the framework of this book, in order to have a common understanding about context and its application in

communications, the concept is introduced considering the existing cited definitions and the context information can be related to entities and the entities can be handling as objects with its respective properties. As a simple concept, the most useful definition is aligned with the most recent context definition, “*the context of an entity is a collection of measured and inferred knowledge that describe the state and environment in which an entity exists or have existed*” [Strassner08].

2.2.2 Context-Awareness/Ubiquitous Computing

Applications taking advantage of environmental information, principally outdoor locations, can be considered as the precursors of the concept of context-awareness [Abowd97]. In fact, the conceptual definition has not evolved too much. In the framework of this book: “*context-awareness is the capability that helps certain applications to exhibit a more personal degree of interaction with the user*” [Abowd97], [Dey97]. More specifically, the certain applications are intending for helping to solve different technological service problems and particularly management problems.

2.2.3 Policy-Based Management

In the field of network management, a policy has been defined as a rule directive that manages and contains the guidelines for how different network and resource elements should behave when some conditions are met [IETF-RFC3198]. In other words, a policy is a directive that is specified to manage certain aspects of desirable or needed behaviour resulting from the interactions of users, applications and existing resources or services [Verma00].

In the framework of this book, an initial definition to consider is “*Policy is a set of rules that are used to manage and control the changing and/or maintaining of the state of one or more managed objects*” [Strassner04], since this definition is more applicable to managing pervasive services and applications. The inclusion of state is important for pervasive systems, as state is the means by which the management systems knows if its goals have been achieved and if the changes that are being made are helping or not.

2.2.4 Pervasive Services

A pervasive service has been defined as a service that takes into account part of the information related to context in order to be offered [Dey00a]. However, as a result of the advent of new devices that are faster, more efficient, and possess greater

processing capabilities, pervasive services have been conceptualized as more device-oriented, and the applications designed for them consider not only single user benefits, but more importantly, the interaction between users and systems. This increases the effect that mobility has on a pervasive service. In the framework of this book, a *pervasive service is one that makes use of advanced ITC mechanisms to facilitate service management operations and manifest itself as always available.*

2.2.5 *Ontology Engineering*

In the framework of this book, ontology is a specification of a set of concepts using a formal language that can define and describe properties, features or relationships between the concepts in a particular domain or even between concepts of different domains (e.g. management networks and services).

The term ontology is often used in a more generic way, as a formal and explicit specification of shared conceptualizations [Gruber93b], [Gruber95]. In other words, ontology is a formal description of the concepts and relationships that can exist for an agent or a community of agents [Guarino95].

This definition is consistent with the usage of ontology in communications, thus an ontology is a standard definition or array of concepts. In terms of a standard/formal language, ontology is a conceptualization representing abstract model(s) that contain concepts that are relevant to a particular domain, management operations of pervasive and autonomic communication domains, virtual infrastructures and/or cloud computing, for example.

2.2.6 *Autonomic Communications*

This section discusses the use of autonomic communications, highlighting the management of information and resources, service re-configurability and deployment, and the self-management requirements inherent in autonomic systems. The purpose of autonomic systems is to solve problems of managing complex service and communications systems [Strassner06c], [IBM05].

Autonomic systems are the result of information technologies interacting and cooperating between them for supporting service management operations (e.g. creation, authoring, customization, deployment and execution).

The interaction is supported by the extra semantics available from context information using ontologies and other information technologies, such as the policy-based paradigm for managing services and networks [Serrano06b]. The interactions are shown in Fig. 2.1.

Autonomic computing is the next step towards increasing self-management in systems and coping with the complexity, heterogeneity, dynamicity and adaptability

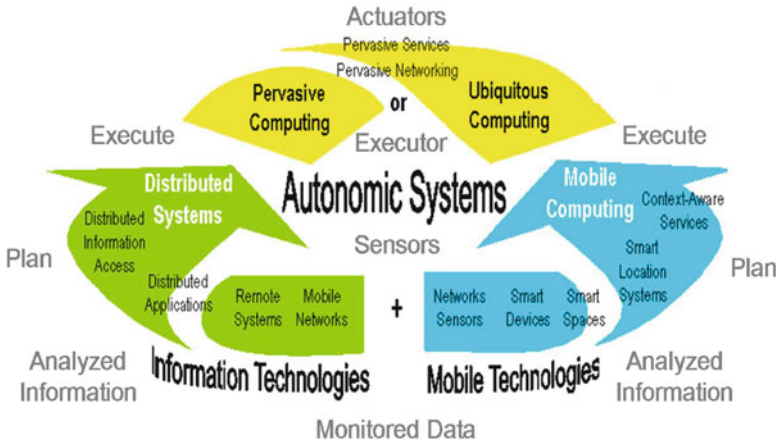


Fig. 2.1 Combining autonomic computing with communication systems and technologies related

required in the modern communication services systems. This trend of increasing complexity is evidenced by the convergence of advanced pervasive systems with engineering technologies (networking, service and knowledge and information technologies) and the Internet capabilities.

Autonomic communications support the idea of autonomic elements, which process the management information to provide the autonomy necessary to support next-generation networks and services.

This section argues that context, which provides increased semantics (and hence, increases the ability to reason about management data), should be used to augment existing management to fulfil the goal of providing self-managing functionality, this is a premise used to understand the autonomic concept in this book. Autonomic elements are to be built to satisfy information models; hence, context should be added as an element of the information model (or at least made available to it). This will not only provide greater functionality, but also enhance the interoperability of the system.

Autonomic services, as much as systems and networks become more self-managing, the nature of the services should evolve accordingly. Thus, the services should be developed based on autonomic computing principles, so that the services also become self-managing.

Autonomic elements, services must be more flexible in order to respond to highly dynamic computing environments and be more autonomous to satisfy the growing and changing requirements from users. This means that services and components (software pieces or devices) need to be more context-aware. The addition of context to information models, augmented by ontological information, is the key to creating services that are self-managing.

Autonomic management, to cope with increasing management complexity, it is necessary that systems support the eight features that characterize an autonomic

system, as defined by IBM [IBM01b]. These eight features have guided the development of autonomic systems. The following subsections review these critical autonomic communications features [IBM01b], [Strassner06c].

2.2.6.1 Self-Awareness

As per definition, self-awareness is the capability of a system to collect pieces of information and process them locally, this is a feature to be aware of its state and its behaviour, with the objective of governing operations itself, and for sharing and collaborating its resources with other systems in a more efficient manner.

2.2.6.2 Self-Configuring

This is the capability of a system for adapting dynamically and automatically to changes in its environmental conditions. Self-configuring can refer firstly to configuring conditions (setup), which must be done largely automatically in order to handle changes in the environment, and secondly to the adaptability of the architecture to re-configure the system for achieving service quality and experience factors.

2.2.6.3 Self-Optimization

This is the capability of a system to improve the resource utilization and workload of the system following the requirements from different services and users. This resource utilization and workload is dependent on the time and service lifecycles for each service and user. Performance monitoring and resources control and optimization are management operations inherent in this process.

2.2.6.4 Self-Healing

This is the capability of a system to detect and prevent problems or potential problems, and then, as a result of this action, to find alternate ways of using resources or reconfiguring the system to avoid system or service interruptions. To perform this capability, local data processing is needed.

2.2.6.5 Self-Protection

This is the capability of a system that defines its ability to anticipate, detect, and protect intrusion or attacks from anywhere. This depends on its ability to identify failures in the system, and enables the system to consistently enforce privacy rules and security policies.

2.2.6.6 Context-Awareness

This is the capability of a system to process pieces of information in their environment, including their surrounding, and activity, with the objective to react to that information changes in an autonomous manner as a result of the utilization of elements and process in its environment.

2.2.6.7 Open

This is the characteristic of a system to operate in heterogeneous environments and across multiple platforms, always using and implementing open standards. Autonomic systems must operate in a heterogeneous world, and hence cannot be implemented using proprietary solutions.

2.2.6.8 Anticipatory

This is the characteristic of a system to be prepared for providing the optimized resources needed in order to seamlessly provide better functionality for its users, all the while keeping the complexity of the system hidden.

It is important to highlight, in this book, that these eight main features for autonomic systems have been related to the most important generic service operation requirements to determine the impact that these features have on service requirements, and to create a vision in how these service operation characteristics can meet the requirement of an emerging areas such as cloud computing.

Figure 2.2 shows that context-awareness is one of the most important features to be considered when supporting services are related to all of the other service operation requirements [Serrano06b]. Thus, its crucial handling and dissemination of the context information support the pervasive services and its feature of context-awareness enables autonomic systems control. Self-configuring is also a requirement that is related to all of the other service operation requirements; however, it is related more to the implementation perspective.

2.2.7 Virtual Infrastructure and Cloud Computing

It is anticipated that cloud computing should reduce cost and time of computing and operations processing [IBM08]. However, while cost benefit is reflected to end user only, from a cloud service provider perspective, cloud computing is more than a simple arrangement of mostly virtual servers, offering the potential of tailored service and theoretically infinite expansion [Head10]. Such a potentially large number of tailored resources which are interacting to facilitate the deployment, adaptation and support of services, this situation represents significant management challenges.

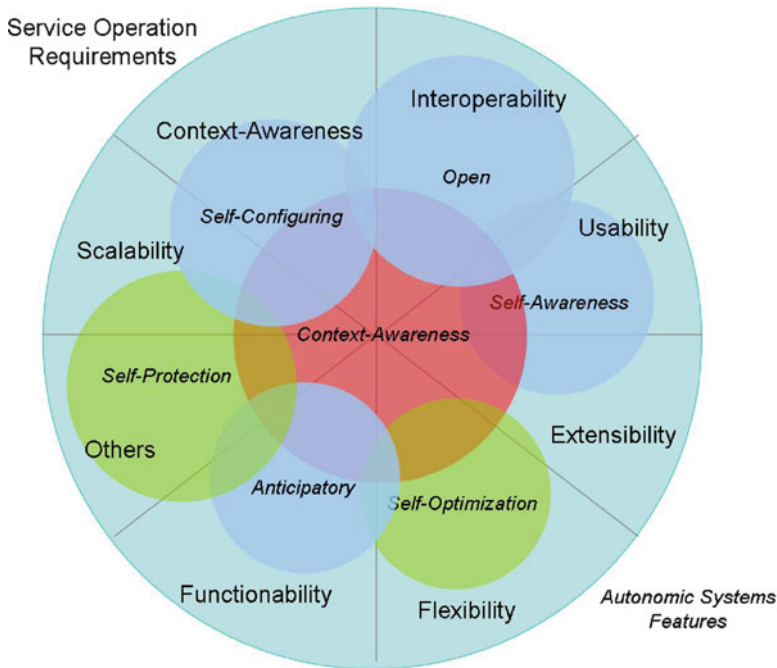


Fig. 2.2 Relationships between autonomic systems features and service operation requirements

In management terms, there is a potential trend to adopt, refine and test traditional management methods to exploit, optimize and automate the management operations of cloud computing infrastructures [Waller11]; however, this is difficult to implement, so designs for management by using new methodologies, techniques and paradigms mainly those related with security are to be investigated.

The evolution of cloud computing has been benchmarked by a well-known evolution in distributed computing systems [IFIP-MNDSWG]. Figure 2.3 depicts this evolution and shows the cloud computing trends passing from a physical to virtual infrastructure usage and from a local to remote computing operations. This evolution towards cloud computing services era is briefly explained in the following sections.

2.2.7.1 Virtualization

Cloud computing is leading the proliferation of new services in the ICT market, where its major success has been to facilitate on-demand service provisioning and enabling the so-called pay-as-you-go provision and usage of resources and server time over the Internet. The user of the cloud services does not own and does not

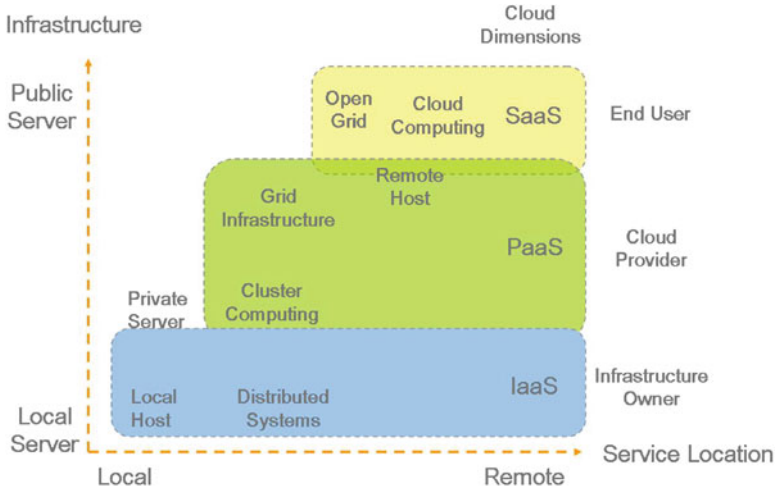


Fig. 2.3 Evolution towards cloud computing solutions

maintain the underlying physical hardware (i.e. servers and network devices or software thereby avoiding additional costs for configuration labour), and pays for permission to use “a virtual slice” of those shared resources.

As today’s cloud computing is known, it has origins in distributed computing systems [Andrews00], [Elmasri00], [Lynch96]. An important feature in distributed systems is the role management systems have in order to control, processes remotely and in a coordinated manner. After a decade of management development, and evolution in computing systems emerges grid computing, a combination of remote computer resources to execute common goals in remote located physical infrastructures [Foster99]. In grid computing no matter where the resources are allocated, tasks are executed in distributed places by using grid managers to pursue multiple tasks even through different administrative domains [Catlett92], [Maozhen05], [Plazczak06], [Buyya09].

In generic terms, grid computing can be seen as a distributed system where large number of non-interactive files is involved. It is important to mention what make grid computing different from cluster computing is that grids seem more seamless coupled, it is definitively heterogeneous, and geographically their allocation is spread out. Grid computing is traditionally dedicated to a specialized application and it is a more common cluster computing which will be used for a variety of different purposes. Likewise grids are more often build with the aim to be used in a more long-term application and involve more use of physical infrastructure resources with specialized or particular ad hoc developed software libraries known as middleware. Examples of middleware are GridWay [GRIDWAY], TeraGrid [TERAGRID], gLite [GLITE], UNICORE [UNICORE] and Globus Toolkit [GLOBUS].

2.2.7.2 Cloud Service

As main feature in cloud computing systems, while labour costs are still present, users pay reduced prices as the infrastructure is offered to and shared by multiple users [Head10], [Urgaonkar10]. According to this model, processing time cost for each user is reduced since it is covered by multiple users, as seen in the traditional pay-for-server-time model [Greenberg09].

It is well accepted by ITC professional that cloud computing is a revolution in the service provisioning and marketing giving an opportunity to bring to bear technological experience and revenue in a new area by exploiting the Internet infrastructure. However, the concept behind this trend is the full exploitation of multitenancy of services, where multiple users can make use of the same infrastructure by using intermediate middleware’s known as virtual infrastructure to use the same information service [Greenberg09].

2.3 Ontology Engineering, Autonomic Computing and Cloud Computing Basis

In telecommunications, pervasive computing applications have always attracted the attention of research and industry communities, mainly because pervasive computing conceptually offers to all of its users (at any level of abstraction, from end user to network operators) broad possibilities for creating/supporting useful and more efficient services.

Pervasive computing requires a mixture of many technologies and software tool/applications to materialize these benefits [Serrano06d], as shown in Fig. 2.4; however, it is context-awareness, the feature that makes the computing applications in pervasive

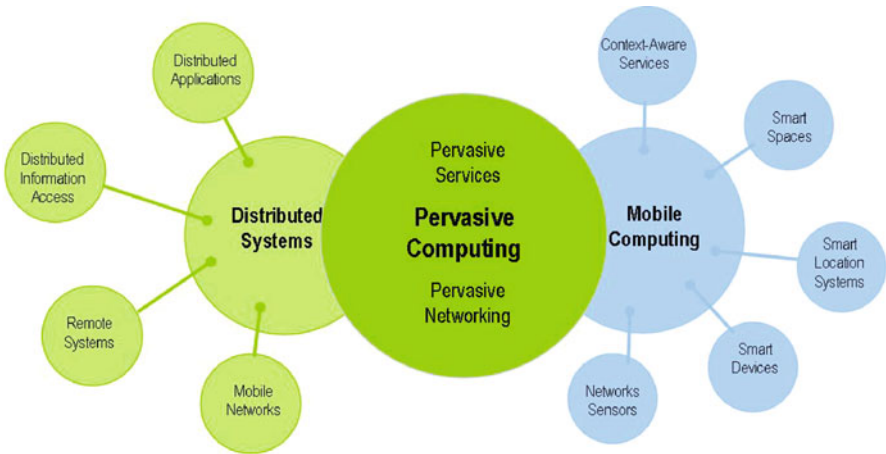


Fig. 2.4 Pervasive computing composition

services more extensible, useful and many times simple. To make this more attractive, this trend will continue, as new mobile and wireless technologies are being integrated.

It is not difficult to imagine that this mixture of technologies increases the complexity of the systems and solutions by many systems and devices that use different mechanisms for generating, sharing and transferring information to each other. Further more existing data models are application-specific and designed independently to each other. In this respect, a method or set of them are needed to enable the efficient and clear exchange and reuse of information between the systems.

This method must be inherently extensible; as systems and networks become more pervasive, the nature of the services provided should be easy to change according to changing context. Services must also become more flexible in order to respond to highly dynamic computing environments and become more autonomous to satisfy the growing and changing requirements from users. In other words, the services must become more adaptive and context-aware.

Simple advances in resources and services have been feasible as a result of the ever-increasing power and growth of associated technologies. However, this drive for more functionality has dramatically increased the complexity of systems—so much that it is now impossible for a human to visualize, much less manage, all of the different operational scenarios that are possible in today's complex systems.

The stovepipe systems that are currently common in OSS (operations support system) and BSS (business support system) designs exemplify this—their desire to incorporate best of breed functionality prohibits the sharing and reuse of common data, and point out the inability of current management systems to address the increase in operational, system, and business complexity [Strassner06b].

Operational and system complexity are induced by the exploitation and introduction of technology to build functionality. The price that has been paid is the increased complexity of system installation, maintenance, (re)configuration and tuning, complicating the administration and usage of the system. Business complexity is also increasing, with end users wanting more functionality and more simplicity.

This requires an increase in intelligence in the system, which defines the need for pervasive applications to incorporate autonomic characteristics and behaviour [Horn01]. Along this book, the assumption that pervasive computing focuses on building up applications using real-time information from different domains is assumed, thus the requirements are that business must be able to drive resources that network(s) can provide.

On the other hand, the aim of this book is to discuss the role of ontology engineering in the cloud era. It is evident from the differences between pervasive applications in one side and autonomic communication in the other. The autonomic systems have been conceived to manage the increasing complexity of systems [IBM01a] as well as to dynamically respond to changes in the managed environment [Strassner06a]. While autonomic communications has proposed some variant of automatic service generation, pervasive applications require a more detailed model of the system that is being reconfigured as well as the surrounding environment. It is due to that autonomic communications is armonized by a standard language where the sharing of information is easier than when formal but different languages are being used as it occurs in pervasive applications.

More importantly, a system for supporting pervasive applications cannot be reconfigured if the system does not “understand” the functionality of the components and any restrictions imposed on that functionality by its users or the environment. This means that the system requires self-knowledge of its components, and knowledge of its users and requirements. In pervasive systems one of the most important forms of knowledge and its most basic stage is context, as it can be used to simply capture knowledge.

An example aiming to clarify the relationship between pervasive applications and autonomic communications is given as follows: let us assume that a pervasive application is able to generate a code to modify performance itself and this is based on variations of contextual information. The model of an autonomic system using context information and its relations is not just the set of self-configuration and other self-* operations; self-knowledge and self-awareness are required in order to provide autonomic behaviour, with the resulting set of decisions based on business rules as well as context-specific information. Since functionality can change, the autonomic solution must incorporate extensible information and data models to accommodate future decision and operations.

2.3.1 Ontologies to Define, Represent and Integrate Context Information

The nature of the information requires a format to represent and express the concepts related to the information. Ontology is an explicit and formal way to capture and integrate information, without ambiguity, so that the information can be reused and shared to achieve interoperability. Explicit means that the types of concepts used, and the constraints on their use, are unambiguously defined. Formal indicates that the specification should be machine readable and computable.

A specific definition of “Ontology,” with respect to system and network management, is a database describing the concepts in a domain, their properties and how the concepts relate to each other. This is slightly different from its definition in philosophy, in which ontology is a systematic explanation of the existence of a concept. System and network management is less concerned with proving the existence of something than in understanding what that entity is, and how it interacts with other entities in the domain [Guarino95].

2.3.2 Using Ontologies for Managing Operations in Pervasive Services

Pervasive computing or ubiquitous computing has evolved and acquired a grade of pervasiveness such that pervasive computing promises to be immersed in everyday activities and environments. Pervasive computing goes beyond the idea that almost

any device, from personal accessories to everyday things, such as clothing, can have embedded computers that can create connections with other devices and networks. The goal of pervasive computing, which combines current advanced electronics with network technologies, wireless computing, voice recognition, Internet capabilities and artificial intelligence, is to create an environment where the connectivity of devices and the information that they provide is always available.

However, in this complex environment where systems are exchanging information transparently using diverse technologies and mechanisms, management becomes increasingly difficult. The increasing multiplicity of computer systems, with the inclusion of mobile computing devices, and with the combination of different networking technologies like WLAN, cellular phone networks, and mobile ad hoc networks, makes even the typical management activity difficult and almost impossible to be done by human beings. Thus, new management techniques and mechanisms must be applied to manage pervasive services.

One of the most important characteristics of knowledge is the ability to share and reuse it. In this context, ontology is used for making ontological *commitments*. An ontological commitment is an agreement to use a vocabulary (i.e. ask queries and make assertions) in a way that is consistent. In other words, it represents the best mapping between the terms in ontology and their meanings. Hence, ontologies can be combined and/or related to each other by defining a set of mappings that define precisely and unambiguously how concepts in one ontology are related to concepts in another ontology. Thus, ontologies are a powerful means to provide the semantic structures necessary to define and represent context information.

Ontologies were created to share and reuse knowledge in an interoperable manner [Guarino95] and to avoid the handicaps founded when different systems try to exchange application-specific heterogeneous representations of knowledge. Such cases are complicated by the difficulties between languages and patois; missing of communication conventions and mismatch of information models.

As a data or information model represents the structure and organization of the data elements, the management activity can obtain benefits from using the data from such elements in its operations. In principle, a data or information model is specific to the application(s) for which it has been used or created. Therefore, the conceptualization and the vocabulary of a data model are not intended a priori to be shared by other applications [DeBruijn03]. Data models, such as databases or XML schemas, typically specify the structure and the integrity of data sets. The semantics of data models often constitute an informal agreement between the developers and the users of such data, and that finds its way into applications that use the data model. By contrast, in the area of knowledge engineering, the semantics of data needs to be standardized in a formal way in order to exchange the data in an interoperable manner [Genesereth91].

Ontologies not only provide enrichment to the information model, but also semantic expressiveness, allowing information exchange between management applications and different management levels. It is this characteristic by which ontologies are emerging into engineering areas, providing advantages to better specify the behaviour of services and management operations. One potential drawback of

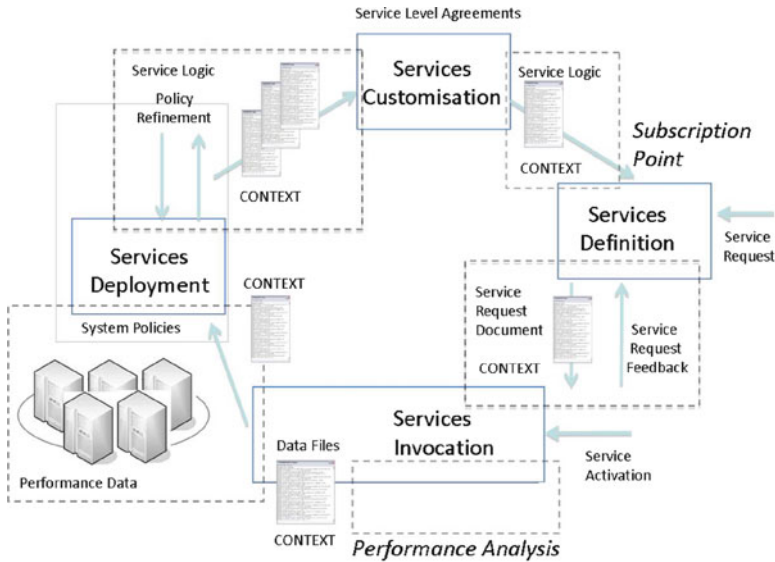


Fig. 2.5 Context information role in pervasive computing environments

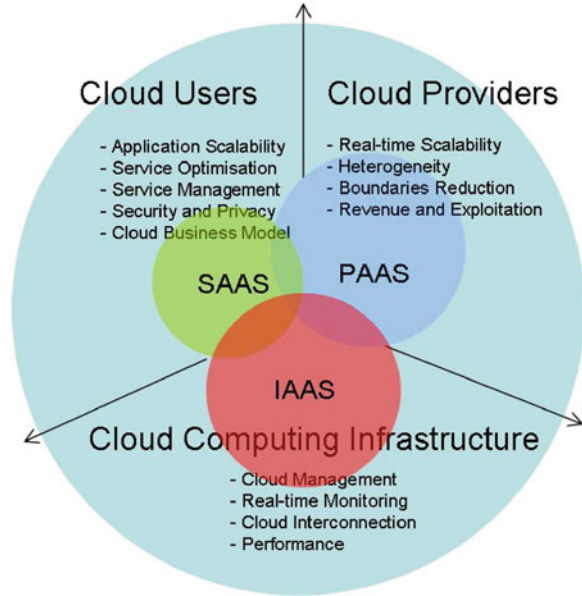
using ontologies for management purposes is that they require significant computational resources. However, the disadvantages are outweighed by the associated benefits of using ontologies. Times for agreements are reduced when a system using ontologies is being used, for instance or even more when the information into the systems need to be shared to other systems, the times for seeking of the information and mapping are reduced.

2.3.3 Autonomic Communications and Ontologies

Autonomic systems emerge as one way to solve management complexity. Complex interactions need to be supported by the increase in semantics embedded in the context information that is represented by ontologies and controlled using policy languages. Autonomic communications can be seen as an approach consisting of the integration of context information in management operations for pervasive services that can be extended into changes in the communication networks by autonomic element executed in the network infrastructures. Specifically, autonomic communications focus on the integration of context information to control management operations in the service lifecycle; to achieve this goal, use the ontology descriptions for management operations and formal modelling techniques.

Figure 2.5 depicts an scenario in which the level of complexity is easily visualized, the context information contained in the networks and devices can be used for multiple operations, even in different domains, and give support for management

Fig. 2.6 Summary of challenges in cloud computing



operations such as customization, definition and deployment and, even more importantly, the service maintenance. In the depicted autonomic environment, the possibility of upload or transfer of context information from its various sources to the management system increases the pervasive level of the applications and the services using such information (shown on the left side of Fig. 2.4).

This level is depicted as a bar in order to indicate that context information must be *translated* between each level. The translation is simplified and automated when formal languages are used. One of the objectives of this book is to explain the need for making context network information available to different service abstraction layers, such as those shown in Fig. 2.4 for triggering appropriate management operations in autonomic systems and also to provide guidance to find out the best possible approaches to achieve this goal.

2.3.4 Cloud Service Challenges and Ontology Engineering

In this section, a number of research challenges for cloud computing is described. It is not intended to be an exhaustive list, but rather to introduce important issues with respect to improving cloud management, and which are yet open issues and under investigation in the emerging cloud computing area. Figure 2.6 summarizes these trends based on well-identified cloud demands [Greenberg09], [DEVCENTRAL].

Figure 2.6 depicts trends and challenges in cloud computing, particularly from a user perspective, where cloud computing is being focused according to IDC

Enterprise Panel, “clients themselves are concerned with Security, Availability and Integration, today, and over the next 3 years, customers will be moving to Cloud Services” [DEVCENTRAL].

2.3.4.1 Real-Time Scalability of the Cloud Service

Cloud services must reflect the current total service load to address time-varying demand. Thus, the application must actively participate in the process to produce changes in the underlying virtual or physical infrastructure. In other words, individual services need not to be adaptive with system load; however, they generate information that can be used as inputs for applications used to modify infrastructure performance. Therefore, the application is not static anymore; in the cloud, it can evolve over time to migrate or expand using more or less computing resources. So cloud services are agnostic of the underlying infrastructure, where possible, and it is the job of the physical infrastructure to host the services in a self-adaptive manner.

2.3.4.2 Scalability of the Application

In cloud computing, adding more resources does not necessarily mean that the performance of the application providing the service will increase accordingly. In some situations, the performance may decrease because of the managing/signalling overhead or bottlenecks. A challenge is to design proper scalable application architectures as well as making performance predictions about data and processing load balancing to satisfy service load demands and prevent future demands.

2.3.4.3 Optimization for Complex Infrastructure and Complex Pricing Model

If well-known pay-as-you-go model makes cloud very attractive, there are many possible other models for application architectures where revenue is the main benefit. A challenge is to design an optimal architecture to fit a cloud environment and at the same time to optimally use all resources with respect to pricing and billing. Note that the pricing model may conflict with the optimally scalable architecture, thus requiring a carefully designed trade-off of application and infrastructure design priorities is the challenge.

2.3.4.4 Cloud Scalability Limitations

Cloud services are commonly referred to as “infinitely” scalable. However, far away from the technology is near to reach this concept. A concept of this dimensions is not generally possible to solve physically by just expanding the server farm, many

systems working cooperating and co-ordinately are needed. It is of interest to enable federation of resources from several cloud providers. This has recently been addressed and emphasized by Google through a statement made by Vint Cerf at the Churchill Club on FORA.tv on the Inercloud and the Future of Computing [FORATV].

2.3.4.5 Maintaining Service Levels

As in networking services, with increasing amount of resources consumed, reduced service levels and failures are commonplace, in other terms the quality of the service is specifically in the case of IaaS, when applications must handle them without any interruption. Networking resources are seen as a particularly critical resource and a common point of failure; hence, QoS is closely related.

2.3.4.6 Security

With the participation of multiple users in the cloud environment, the privacy and security of the information being stored, transmitted or operated is crucial. In public clouds, this issue is still driving research activity in terms of efficiency, data protection algorithms and protocols [Mace11]. This is motivated mainly by legal issues about security requirements for keeping data and applications operating in a particular legal domain (e.g. Ireland, EU, etc.) and not due to technological constraints.

2.3.4.7 Heterogeneity of Cloud Platforms

There are multiple vendor-specific platforms—the possibility to build applications or services that are provider-neutral to prevent lock-in would allow redundant deployments, reduce staff training costs (and OPEX in general), and would also enable best of breed (or cost optimal) service provider selection. Close relation exists between the development of adequate and efficient management tools and/or systems managing the multiplicity [Rochwerger11]. Up-to-date cloud service provisioning has been the central activity around this area; however, once the cloud will achieve a certain level of maturity, the development of inter-connected platforms for the cloud will be a necessity.

2.3.4.8 Management of Underlying Resources—Transparency

In cloud computing, where underlying resources are hidden/abstracted, service management remains like an area where research efforts are required to find the best methods and mechanisms to enable services to be managed externally.

Today the cloud and its underlying infrastructure is managed in the form that services capabilities are not exposed [Srikanth09]. The cloud computing paradigm establishes that computing operations and service applications can be performed at unlimited scalable levels and following on-demand service attention, where (at least from a service end-user perspective) the less important is to worry about infrastructure. However, in PaaS and IaaS levels, a key feature for the cloud optimization and at the same time increase the performance of the cloud infrastructure is that the low-level management of underlying resources and services remains hidden from the application, and the end user has to relieve the platforms and management user, when it is needed, to manage them. However, the end user still requires some flexibility in terms of the resources and service they pay for so [Sedaghat11]. A key challenge remains in the mapping of high-level user requirements to low-level configuration actions. It is also vitally important that these management actions can be performed in a scalable manner since the cloud service will not be able to individually configure a large number of shared resources and services for individual users.

2.3.4.9 Management Scalability Towards Federation

In cloud computing, if the number of partitioned virtualized resources (or instances) grows the challenge of how to control resources individually become more apparent. This can be addressed by allowing users to manage their own delegated resources—non-expert resource management. This raises a number of challenges including: support non-expert monitoring and management, managed delegation of management resources and authorities, and auditing [Goiri10]. This would mean that users could monitor their own resources so they can decide when to request more resources and ensure they are getting a near optimal value for the resources they pay for (depending on the payment model used).

2.4 Identifying Requirements for IT Infrastructures and Cloud Service Management

The identification of service requirements, which is between others one of the main research tasks of this book, is presented in this section. This section is divided into four parts: (1) information requirements, (2) end-user requirements, (3) technology requirements and (4) rather to present an exhaustive list of cloud computing challenges and service requirements, it concentrates in the most important up-to-date challenges and where this emerging area has advanced developing tool and technologies to achieve cloud computing level has up to date.

The considerations presented in this section are based on exhaustive study, analysis and discussions about the state of the art in IT and cloud systems and on technical experience from research activities. Basic conceptual frameworks applied

developing technologies through participation in international European projects and research network of excellence. Also public documentation that guides the evaluative research tasks to define the state of the art of context-aware services in the pervasive computing knowledge area is available in [IST-CONTEXT], [SFIFAME], [AUTOI], and [IST-EMANICS]. This section acts as a platform research work towards a conceptual framework based on integration to define solid basis and technological principles for supporting pervasive services in cloud environments.

Particularly, pervasive services are often related to mobile devices, because in such cases the location of the device or user, or the type of the user's connectivity, is changing. These factors emphasize one of the most important characteristics of pervasive services: their ability to adapt their service (most likely in as seamless manner as possible) to these changes, and so facilitate truly mobile applications. Pervasive services can also be useful for non-mobile users or users who have a fixed connection or a set of network applications.

For example, the environment of a user may also change and, in such case, the user experience and rapid service re-provisioning can also be enhanced by pervasive applications. Changes in the network conditions can be performed by the pervasive service and the user is insulated from those changes as a result of pervasive service acting.

It is important to emphasize the context-aware capabilities of pervasive services which enable new types of applications in pervasive computing giving the opportunity to built based on heterogeneous technologies, platforms offering inter-connected services. These applications, for example can help users to find their way in unfamiliar areas, receive messages in the most useful and suitable manner, or find compatible people. (A traditional example for pervasive service applications.) All of these and other activities require the system and its devices to be managed in an autonomic manner. This is because the use of context information in pervasive applications reduces the amount of human intervention that an application needs for satisfying user's requests.

The current growing market of middleware applications is also focused on providing applications, from different levels of abstraction (e.g. business, architecture, or even programming points of view), the ability to share information and thereby deliver relevant context information. These middleware applications result in an increased number and development of applications that are able to use context information in order to improve the services that they deliver [Brown97]. In other words, context information is not just user-oriented, as in the beginning when the concept of context-awareness was first introduced, but it is also related to the management of different services and applications. It is only recently that, with the intervention of pervasive computing, more applications using context information have started to be developed as services with context-aware properties. Context-aware applications can enhance the capabilities of devices and applications, and such services are known as pervasive services. For example, in [Abowd97], applications using context information for different tasks in the ubiquitous and services areas are described, while in [Tennenhouse97], there are applications using context for networking purposes.

2.4.1 *The Information Requirements*

One of the most difficult aspects of context information is its dynamism. Changes in the environment must be detected in real time, and the applications must quickly adapt to such changes [Dey01]. The nature of the information is the most important feature of context-aware applications and systems; if pervasive applications can fully exploit the richness of context information, service management will be dramatically simplified [Brown98]. Other important challenges of pervasive services include: (1) how to represent and standardize the context information, (2) if the information is correctly represented, and (3) if the information can be translated to a standard format, then different applications can all use the information. Finally, some types of context also depend on user interfaces (which can make retrieving context information easier), or the type of technologies used to generate the context information. This sets the stage for discussing the information requirements in this section.

2.4.1.1 **Modelling of Information—Context Information Modelling**

Modelling context information is one of the major challenges when services deployment design is needed [Krause05]. Without a well defined, clear and at same time flexible information model, applications will not be able to use such information in an efficient way for taking advantage of all of the benefits that the context information can provide for the service as well as for the provisioning of that service. The context information model must be rich and flexible enough to accommodate not only the current facets of context information, but also future requirements and/or changes [Dey01]. It has to be based on standards as much as possible and moreover, the model should scale well with respect to the network and the applications. This introduces a great challenge managing this context information in a consistent and coherent manner. Storage and retrieval of this information is also important. A most well-established approach is to model context information model representation and the framework proposed to represent the context information supported by an object-oriented, entity-centred model.

Figure 2.7 shows an entity-centred model representation. The model is based on simple concepts and its relationships, as syntactical descriptions, between those concepts. An entity is composed of a set of intrinsic characteristics or attributes that define the entity itself, plus a set of relationships with other entities that partially describe how it interacts with those entities.

The entities can represent anything that is relevant to the management domain [Chen76]. Moreover, the relations that can exist between the different model entities can represent many different types of influence, dependence, links and so on, depending mainly on the type of entities that these relationships connect. The model's objective is to describe the entity and its interaction with other entities by describing the data and relationships that are used in as much detail

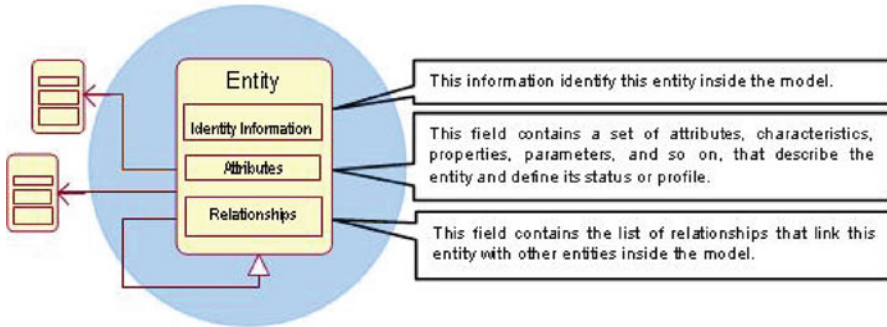


Fig. 2.7 Context model template

as is required. This abstraction enables the model to be made more comprehensible by different applications. Since this format is machine readable, the information can be processed by applications much easier than an equivalent, free-form textual description.

The entity model can be thought as a general purpose way of representing and storing context information throughout the network. Modelling context is a complex task, so using an extensible model provides a “template” that both standardizes the information and enables it to scale its contents for future applications. This entity-centred model can be used to characterize context information.

This model is inherently extensible, and new attributes or relationships can be added to entities without having to modify the structure of the model. The model must be reused, as any new entities that are required can simply be added to the model and linked to the existing entities by defining appropriate (new) relationships, without having to modify any of the existing entities. The entity-centred model is easily scalable. Along this book multiple references which use the entity-relationship paradigm for modelling context information can be found, the reason is simple, by using this model it results in an easy and extensible management mechanism to represent, handle and operate information.

2.4.1.2 Mapping of Information—Context Information Mapping

The entity model provides a powerful abstraction of the information needed by context-aware applications and, in general, the pervasiveness required by such applications. The mapping of context information can be seen as a distributed data base model, where the entities contain only their own information and also the type of relationships with other data entities. This enables an entity to use the attributes of these other entities if needed. This method of representation acts as a suitable scenario description without reference to any specific element or object, and hence is applicable to many different applications. The mapping of the entity model can be thought of as a general purpose way of represent; store and exchange context

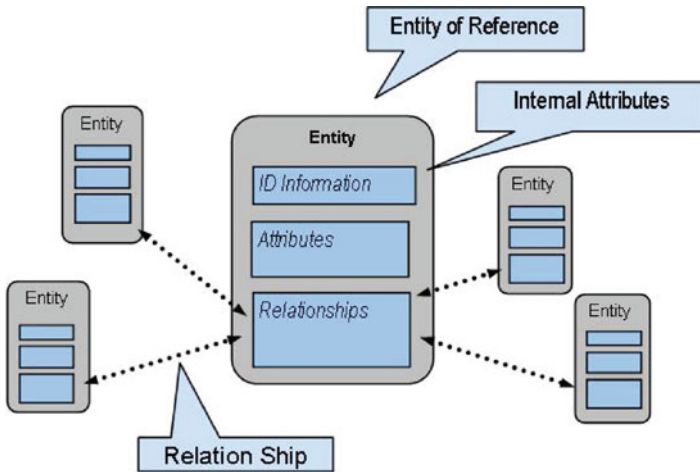


Fig. 2.8 Entity internal architecture

information throughout the network, and for this reason, this concept is used to model context information for mapping purposes.

An abstract model that can be adopted and when necessary add new information to the model is the most appealing solution needed, as a clue for generating this type of models, it is not necessary to modify the existing entities; all that is required is to create a new entity and establish suitable relationships with existing entities. This ensures the scalability of the information model.

Definition of Main Entity Classes

Figure 2.8 shows the architecture of the entity model used and supported in this book. This model defines an object-oriented taxonomy of names required for management, such as Persons, Objects, Places and Tasks. Every entity in the final model will be an instance of one of these generic entity classes. A model for the relationships shall be generated as well, which defines the classes of relationships that must exist (and their attributes), in order to identify different possible relationships between entities and what they mean. The traditional entity definition is described in [Chen76].

Definition of Relationships Classes

The main entities must be related or linked with other entities, of the same or different class, by means of different types of relationships. These relationships also provide useful context information. Since relationships are so powerful, a set of different

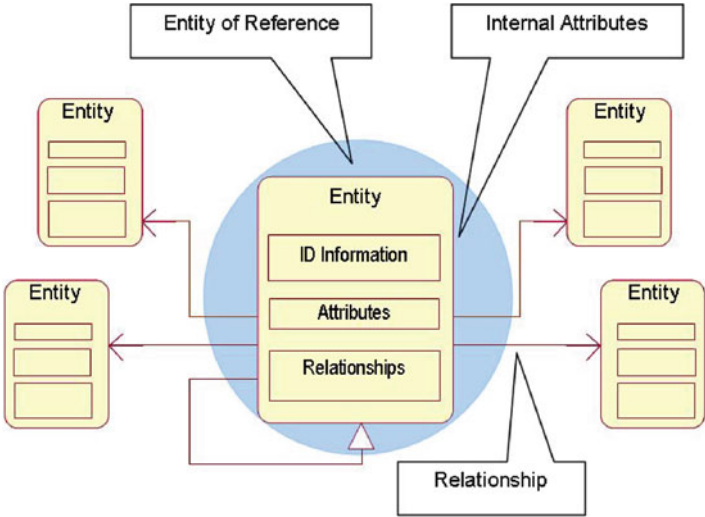


Fig. 2.9 Main relationships between entities

classes of relationships that express the different types of interaction between the different types of entities are defined. Figure 2.9 represents an example set of initial relationships that could be established between the entities of the model. These are high-level type of associations between the four main types of entities. Note that certain types of relationships are only meaningful between certain entities, for example social relations only have meaning between entities of person class and not, for example between places.

2.4.1.3 Taxonomy of Information—Context Information Taxonomy

The classification of context information is not an easy task, due to its extreme heterogeneity. Therefore, this taxonomy could be defined in multiple ways and from multiple perspectives. To identify the information that could be relevant to pervasive applications, the different management operations required by pervasive services were classified based on an extensive literature survey that was conducted to understand the current state of the art in context-aware service provisioning. In this section, a review of different kinds of classifications (most of them orthogonal and compatible) with pervasive services and management operations is presented, a more detailed description can be found in a previous work on [Serrano05]. In a first approximation, context information can be classified by the following characteristics:

By its persistence:

Permanent (no updating needed): Context, which does not evolve in time, that remains constant for the length of its existence (e.g. name, ID card), or Temporary

(needs updating): Context information that does not remain constant (e.g. position, health, router interface load).

By its medium:

Physical (measurable): Context information that is tangible, such as geographical position, network resources, and temperature (it is likely that this kind of information will be measured by sensors spread all over the network), or

Immaterial (non-measurable by means of physical magnitudes): Other context information, such as name or hobbies (it is likely that this kind of information will be introduced by the user or customer themselves).

By its relevance to the service:

Necessary: Context information that must be retrieved for a specific service to run properly, or

Optional: Context information which, although it is not necessary, could be useful for better service performance or completeness.

By its temporal characteristics:

Static: Context information that does not change very quickly, such as temperature of a day, or

Dynamic: Context information that changes quickly, such as a person's position who is driving.

By its temporal situation:

Past: Context information that took place in the past, such as an appointment for yesterday, which can be thought of as a context history, or

Present: Context information that describes where an entity is at this particular moment, or

Future: Context information that had been scheduled and stored previously for future actions, such as a meeting that has not yet occurred.

Based on the above context information taxonomy, the state-of-the-art survey, the operator's expectations, typical scenario/application descriptions and the fundamental requirements for provisioning context-aware network services, Fig. 2.10 shows a representation of the context information taxonomy referred to and described above.

2.4.1.4 Representation of Information—Context Information Representation

Context modelling depends on the point of view of the context definition and scope. The model is a first approximation on how to structure, express and organize the context information as it is defined in [Dey00a] and [Dey01]. As mentioned before,

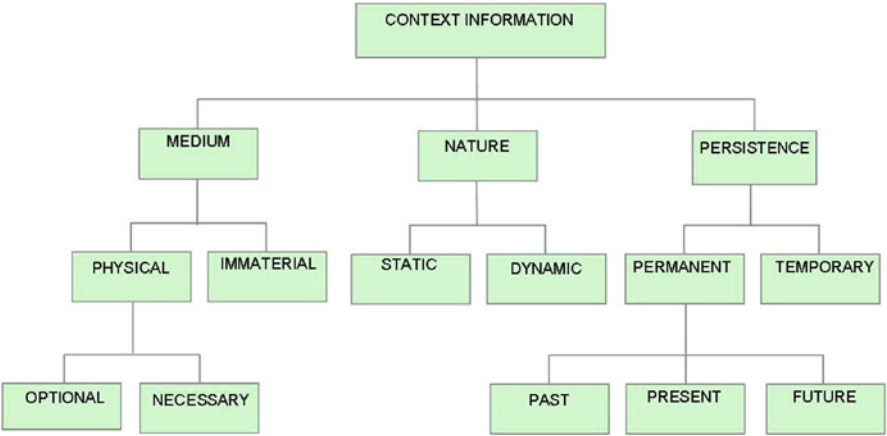


Fig. 2.10 Context information data model

the model is based on the concepts of entity and relationship and derived from the definition of entity in [Chen76].

An entity is composed of a set of intrinsic characteristics or attributes that define the entity itself, plus a set of relationships that are each instances of a standard set of relationship types. The concept of the local context of an entity can be defined as the information that characterizes the status of the entity. This status is made up of its attributes and its relationships. Moreover, the relationships that can exist between the different entities inside the model, as well as the entities themselves, can represent many different types of influences, dependencies, and so on, depending on the type of entities that these relationships connect.

With this type of model, one can construct a net of entities and relationships representing the world surrounding the activity of a context-aware service and thus the models can influence the development of the activity or service. This enables a scenario made up of many different types of information, and the influences or nexus that links one with the others. The local context enables the service to select and use context information from this scenario that is considered relevant in order to perform its task and deploy its service.

Figure 2.11 shows a simple, high-level example of modelling context by means of this entity-relationship technique [Serrano05]. In this example, two possible entities inside a hypothetical scenario are defined as a reference (an entity that represents a person and an entity that represents a printer device). The local context is defined as the sum of all of the specific attributes, plus the relationships established with other entities inside the model, of these two entities. Hence, this figure synthesizes the concept of local context of Person 1 or Printer 1, neither of which include entities such as Person 2 or Person 3. An entity becomes to be a part of local context in other entity when a relationship is defined, so Person 1 can be a part of local context of Printer 1 (right hand-side circle on the figure) and at the

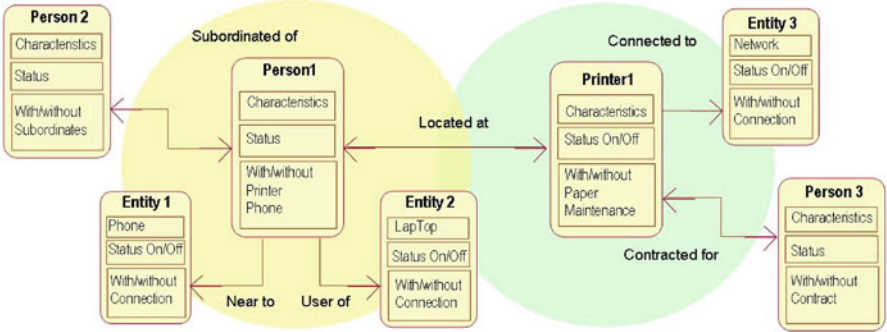


Fig. 2.11 High-level entity model with relationships example

same form Printer 1 can be part of local context of Person 1 (left hand-side circle on the figure).

2.4.1.5 Representation Tools for Information—Context Representation Tools

The tools that could be used to represent and implement the model, and the way to integrate this information model inside the general context-aware system architecture, need to be identified and tested, as they are potential tools for representing the context information. XML is a flexible and platform-independent tool that can be used in different stages of the information representation, which makes implementation consistent and much easier. The use of XML is increasing every day; however, it is by definition generic. Therefore, new languages that are based on XML have been developed that add application-specific features as part of the language definition. For example, to customize services, languages must have concepts that are related to the operational mechanisms of that service. It is in this context that XML Language has been successfully proposed and used to represent the context information models. XML has the following advantages:

XML is a mark-up language for documents containing structured information. The use of XSD (XML schema definition) facilitates the validation of the documents created, even in a more basic but in some way also functional the use of DTD (document type definition) is also an alternative for validation. This validation can be implemented in a JAVA program, which can be the same used for creating and maintaining these XML schemas and/or documents.

Use XQuery, as a powerful search engine, to find specific context information inside the XML documents that contain all the information related to a specific entity. These queries can select whole documents or sub-trees that match conditions defined on document content and structure.

An example of using XML language for describing context information is shown in Fig. 2.12. This represents the context information model (CTXIM) as an example.

Fig. 2.12 XML information model representation

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Information Model for context in
pervasive environments.
Author: Martin Serrano -->
<CTXTIM xmlns:xsi="Context Information
Model" xsi:noNam >
  <TimeStamp/>
  <ValidityPeriod/>
  <Person>
    <FingerPrints>
      <PersonIdentity/>
      <Profession/>
    </FingerPrints>
    <Preferences>
      <Interface/>
      <Service/>
    </Preferences>
  </Person>
  <Place>
    <Geographical/>
    <PhysicalSourround/>
  </Place>
  <Environment>
    <Time/>
    <Date/>
  </Environment>
  <Task/>
  <Object_Device>
    <Application/>
    <Network/>
    <Resource/>
  </Object_Device>
</CTXTIM>
```

Person, Place and Task entities are contained with specific descriptions as part of each entity.

2.4.1.6 Formalization of Context Information—Application of Ontologies

Context-awareness requires the following question to be solved: how can the context information be gathered and shared among the applications that use it? The answer to this question requires an extensible and expressive information model. The format to contain the information is part of the modeling process and the methodology to create the model. The most important challenge is to define the structure of the context information to collect, gather and store information. Context information can be used not just to model information in services, but also to manage the services provided. The model must be rich in semantic expressiveness and flexible enough to consider the variations of current status of the object being managed [McCarthy93]. The model should scale well with the network or the application domain.

A model considered reference in this book, about context modelling in pervasive computing environments, can be found and explained in [McCarthy97]. In this book the modelling result from this previous analysis and modelling activity formalizing

information is referred as an excellent example. In other words and aligned with the objective of demonstrative facts pursued in this book, if the information models are expressive enough, pervasive systems can use that information to provide better management service operations. In order to formalize the information contained in the information model, ontologies appear to be a suitable alternative. However, this does not mean that other approaches are unsuitable for different applications. In this section, the idea to be discussed is that ontologies, in the field of management services, appear as a suitable alternative to solve the problem of formal modelling identified as providing the required semantics to augment the data contained in the information model in order to support service management operations. Ontology engineering can act as the mechanism for formalizing the information and provides the information with the semantic and format features, as it is the main scope in this section.

2.4.1.7 Distribution and Storage for Information Model—Context Model

The nature of context information is forcing systems developers to be able to provide computing facilities, such as context information data models, to anybody, anywhere, at any time. To provide these services, the infrastructures must be developed in a way that allows application programs running in the environment to efficiently locate the services. Currently, due to the numerous different applications that each use a different mix of technologies, each application tends to develop its own specific mechanisms to manage, store and process context information. In this section, an analysis based on generic approaches following their characteristics is briefly described.

Hierarchical Model

This organizes the data in a tree structure. This structure implies that a record can have repeating information, as each parent can have multiple children, but each child can only have one parent, and it collects all the instances of a specific record together as a record type. It is a clear example of dependency in higher hierarchical levels.

Network Model

This model organizes data as a lattice, where each element can have multiple parent and child records. This structure enables a more natural model for certain types of relationships, such as many-to-many relationships.

Relational Model

This model organizes data as a collection of predicates. The content is described by a set of relations, one per predicate variable, and forms a logic model such that all predicates are satisfied. The main difference between this organization and the

above two is that it provides a declarative interface for querying the contents of the database.

Object/Relational Model

This model adds object-oriented concepts, such as classes and inheritance, to the model, and both the content and the query language directly support these object-oriented features. This model offers new object storage capabilities to the relational systems at the core of modern information systems that integrate management of traditional fielded data, complex objects such as time-series and geospatial data and diverse binary media such as audio, video, images and applets for instance. It also enables custom datatypes and methods to be defined.

Object-Oriented Model

This model adds database functionality to object programming languages. Information is represented as objects and extends object-oriented programming language with persistent data, concurrency control and other database features. A major benefit of this approach is the unification of the application and database development into a seamless language environment. This model is beneficial when objects are used to represent complex business objects that must be processed as atomic objects. As an example, object-oriented models extend the semantics of the C++, Smalltalk and Java object programming languages to provide full-featured database programming capability, while retaining native language compatibility.

Semi-structured Model

In this data model, the information that is normally associated with a schema is contained within the data, which is sometimes called “self-describing.” In such a system, there is no clear separation between the data and the schema, and the degree to which it is structured depends on the application. In some forms of semi-structured models there is no separate schema, the schema itself can be converted if the data model is previously defined. In others models, the schema exist but only place loose constraints on the data. Semi-structured data is naturally modelled in terms of graphs.

Associative Model

The associative model uses two types of objects, entities and associations. Entities are things that have discrete, independent existence. Associations are things whose existence depends on one or more other things, such that if any of those things ceases to exist, then the thing itself ceases to exist or becomes meaningless.

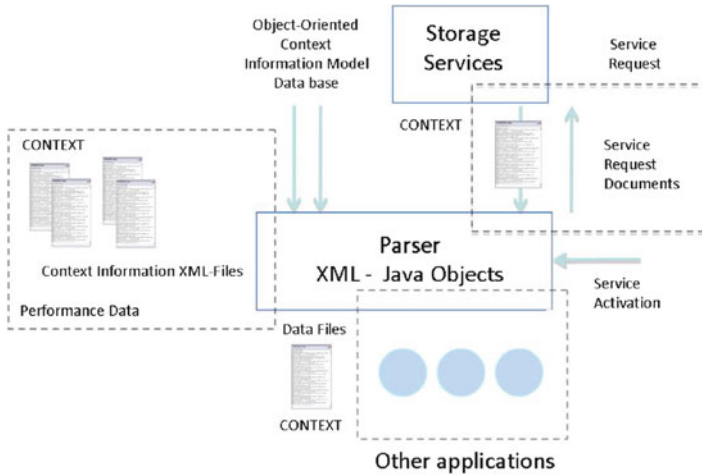


Fig. 2.13 Context information data model

Entity-Attribute-Value Model

The best way to understand the rationale of entity-attribute-value (EAV) design is to understand row modelling, of which EAV is a generalized form. Consider a supermarket database that must manage thousands of products and brands, many of which have a transitory existence. Here, it is intuitively obvious that product names should not be hard-coded as names of columns in tables. Instead, one stores product descriptions in a products table: purchases/sales of individual items are recorded in other tables as separate rows with a product ID referencing this table. Conceptually, an EAV design involves a single table with three columns, an entity, an attribute, and a value for the attribute. In EAV design, one row stores a single fact. In a conventional table that has one column per attribute, by contrast, one row stores a set of facts. EAV design is appropriate when the number of parameters that potentially apply to an entity is vastly more than those that actually apply to an individual entity.

In this section, different types of information databases have been studied. The context model is an alternative that seeks to combine the best features of different models; however, because of this combination, the context model is a challenge to implement. Thus, the possibility to use other data models is open. An object-oriented model, as shown in Fig. 2.13, has been implemented in Java. Extended explanation can be found in [Serrano06d]. The fundamental unit of information storage of the context model implemented in this book is, as described above, a Class, which contains Methods and describes Objects.

A consolidated context data model must combine features of some of the above models. It can be considered as a collection of object-oriented, network and semi-structured models. To create a more flexible model, the fundamental unit of information storage of the context model is a Class. A Class contains Methods and describes an Object. The Object contains Fields and Properties.

A Field may be composite; in this case, the Field contains Sub-Fields. A Property is a set of Fields that belongs to a particular Object, similar to an EAV database. In other words, Fields are a permanent part of the Object, and Properties define its variable part. The header of the Class contains the definition of the internal structure of the Object, which includes the description of each Field, such as their type, length, attributes and name. The context data model has a set of pre-defined types, but can also support user-defined types. The pre-defined types include not only character strings, texts and digits but also pointers (references) and aggregate types (structures).

2.4.1.8 Management of Information—Application of a Methodology

In this book, the use of policies is fundamental to understand the interaction and convergence between different information in different domains, the form of a policy is expressed as follows:

IF <conditions or events> THEN <x-actions> ELSE <y-actions>

Policies are used to manage the service logic at a higher level. A policy is used to define a choice in the behaviour of a pervasive service, and a pervasive service itself comprises a policy-based management system (PBMS). The adaptability of the PBM paradigm comes from its awareness of the operation environment, that is the context in which the management system and its components are being used.

A requirement for the management of information is the capability of the system offering service adaptation. When service adaptation occurs, there is an interesting change in the context. So a pervasive service management information model, which must be in reality an open, vendor-neutral approach to the challenge of technology change, represents an approach with a type of policy information model that also requires a language.

An interesting and extended proposal is to be implemented by using the model-driven architecture (MDA) initiative of the object management group (OMG) [OMG-MDA]. As requirement, a policy-based pervasive service specification language will also need to be used. The PBM methodology is to be implemented and demonstrated as a service composition platform and a service execution environment, both based on open source software and open standards. A set of service-centric network application programming interfaces (APIs) are developed by reusing existing user interfaces (UIs) to hide the heterogeneity of multiple types of end-user access networks and the core networks.

2.4.1.9 Implementation Tools

As far as PBM is concerned, this book do not try to develop new management techniques; rather, the research work is to apply existing PBM techniques to the network aspect of managing pervasive services in cloud environments. In this respect, it is wise to use the background of other standard-based information models and extend

them to satisfy the services requirement needs (e.g. policy core information model (PCIM) [IETF-RFC3060] and [IETF-RFC3460] by the IETF, the core information model (CIM) by the DMTF [DMTF-DSP0005], and the Parlay policy information management (PPIM) APIs by the Parlay Group [Hull04]) between others.

In this book, specific emphasis on the policy-based descriptions for managing pervasive services is given, and a practically functioning pervasive service information model using ontologies to do so is pursued. The fact that there is not any reference implementation for the IETF PCIM [IETF-RFC3060], [IETF-RFC3460] or the DMTF CIM [DMTF-CIM] makes PBM hard to implement using these specifications. Hence, an step further in this section is to support the idea to enhance and control the full service life cycle by means of policies. In addition, this approach takes into account the variation in context information, and relates those variations to changes in the service operation and performance inspired from autonomic management principles and its application in cloud environments.

2.4.2 The User Requirements

This section describes the envisaged requirements for services from each of the parties involved in the service value-chain. The benefits can be obtained for each of these stakeholders with the introduction of context information models that represent the effective functionality of pervasive services in a vendor- and technology-neutral format as summarized. This section refers to work undertaken by the author as collaboration research activities into the EU IST-CONTEXT project and the EMANICS Research Network; the public documentation that defines the state of the art of context-aware services in the pervasive computing knowledge area can be found in the Web sites [IST-CONTEXT] and [IST-EMANICS]. Here is presented just a resume of those main user requirements since the scope that they are necessities to any pervasive service being managed by policy-based managed systems.

2.4.2.1 End-User Requirements

- Cheap end-user devices that provide compelling new functionality.
- Device manufacturer independence.
- Selection from an extensive range of new services.
- Personalized/smarter services

2.4.2.2 Service Providers Requirements

- Personalized services allowing customization of respective service portfolios.
- Significant technological and operational savings as a result of standardization of context information models.
- Significant added revenue from context-aware services.

2.4.2.3 Content Providers Requirements

- Exploitation of context information resulting from common platforms using standardized context information models.
- Flexible entering of partnerships facilitated through standardized context information models.
- The freedom to use any particular service provider for the advertising and providing of content.

2.4.2.4 Network Providers Requirements

- Significant technological and operational savings by using standardized context information models.
- Common equipment requirements and support for CAS based on standard context information models.
- New business opportunities that result from context-aware services.

2.4.3 *The Technology Requirements*

The tendency in modern ITC systems is that all control and management operations are automated. This can be effectively and efficiently driven by using context information to adapt, modify and change the services operation and management offered by their organizations. The context-awareness property necessarily implies the definition of a variety of information required to operate services in next-generation networks.

This section relates and describes the technological requirements to support such ITC context-aware features and identifies the properties of pervasive applications for supporting service management operations in cloud environments. The pervasive properties describe how management systems use context information for cross-layer environments (interoperability) in NGNs to facilitate information interoperability between different service stake holders (federation). The analysis about using information following such technological requirements is a task of this book to exemplify and help as reference to satisfy one or various cloud service management aspects.

The priority of this section is the support of multiple and diverse services running in NGNs. Such scenario(s) are typified by complex and distributed applications, which in terms of implementation and resource deployment, represent a high management cost due to the technology-specific dependencies that are used by each different application. This in turn makes integration very difficult and complex.

Before providing a definition of technology requirements, Fig. 2.12 shows the information requirements for supporting services. This shows the perspective of technology requirements, their relationships and the level of influence between the

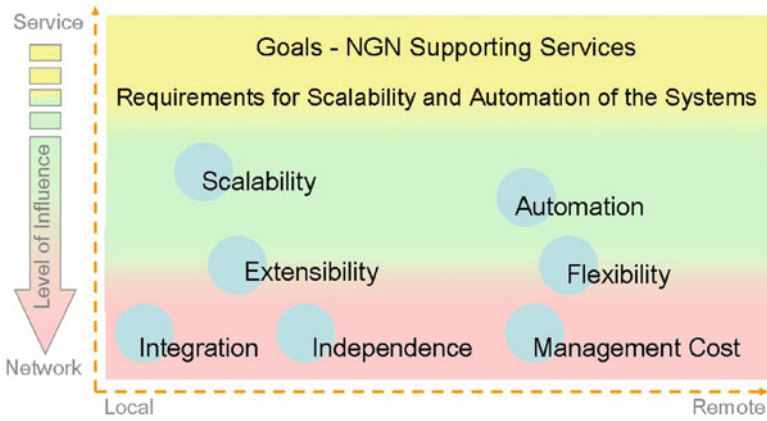


Fig. 2.14 NGN supporting services, relationships and influence

properties that enable the advantages of using context information to be realized by management systems.

Figure 2.14 depicts the level of influence of NGN requirements supporting information services, and how those are related to the service and network views, respectively. On the left hand-side, the arrow shows how the information collected from top-service layers can be used to modify network conditions affecting the different technology requirements of the information management systems.

This section surveys the envisaged requirements for NGNs supporting information services [Serrano06b]. Those requirements are related to management systems supporting services and using integrated information. The requirements are related to the scalability and automation of the systems. These properties will produce increased extensibility and flexibility, as well as produce attendant network and service management cost reductions. They also provide implementation advantages for services as follows.

2.4.3.1 Scalability

Information systems solutions need to scale with the number of users, services and complexity in acquiring and filtering the context information used by the pervasive application. Information needs to be transparently distributed among and along the applications within heterogeneous service environments. The scalability requirements are:

- High levels of scalability due to the inherent necessity to represent diverse types of information within multiple services.
- The necessity to extend the definition and representation of information to other platforms supporting different services.
- Solutions that scale according to the number of users and services in acquiring and filtering information.

2.4.3.2 Extensibility

Extensibility determines the possibility that an information model could be applicable, with the appropriate adaptation, to future or different coexisting applications for pervasive services. The extensibility requirements are:

- Extensibility is required to represent context information and context-aware services using different technologies, platforms and languages.
- Extensibility means that the fundamental structure of the model can accommodate new data and relationships without requiring extensive redesign.

2.4.3.3 Automation

Automating services aims to provide tools and mechanisms for automatic service creation, operation and management. This enables the system to reduce or suppress the need for any human intervention. Information models, supported by updating data mechanisms, can significantly contribute to reach this goal. This implies:

- Mechanisms that use information for automatic service creation, operation and management of the information and services must be supported by the information model.
- Reduce as much as possible the need of any human intervention to manage services.
- Software solutions for supporting self-* operations in terms of diverse user-centred services.

2.4.3.4 Flexibility

Flexibility is the characteristic or capability of the systems for adapting the service creation, deployment, customization and operation to market and user needs. The information model can contribute to this service characteristic by providing the necessary abstractions of the underlying network services and network infrastructure to the entities and stakeholders involved in the service lifecycle. The requirements of this feature are:

- The capability for adapting the service lifecycle management operations to market and user needs must be able to be represented in the information model.
- The necessary abstraction mechanisms to represent the underlying network resources and services must be able to be represented in the information model.
- Middleware solutions that enable business goals to determine information-specific network services and resources must be able to easily use data in the information model.

2.4.3.5 Integration

As a result of new services being defined by independent providers, rather than by the segmentation of specific markets, fragmented approaches using different types of services are offered by different providers using different implementation techniques in different infrastructures. The danger is that the interaction desired by end users is no longer possible due to this fragmentation. A single extensible information model is required that can harmonize the different application-specific services and context information of the different providers. The requirements of this feature are:

- Adaptation of the information systems for providing the information with a format according to the information model. As a result of adaptation, new technologies can support information services from different providers with different implementation mechanisms.
- An extensible information model to enable integration with systems that are designed in a stovepipe fashion.

2.4.3.6 Independence

It is well known that vendor independence is a feature desired for all systems. Hence, the requirements for this feature are:

- Service providers need independence from equipment manufacturers, which promotes the interoperability of the information that they supply for supporting services.
- The information model must be able to model functionality from different vendors that operate on different platforms and use different technologies.

2.4.3.7 Management Cost

Telecommunication services are only viable if they do not incur recurring capital and operational expenditures. The additional traffic and data heterogeneity caused by application-specific information has, up to now, increased operational expenditure, since it requires custom middleware or mediation software to be built to integrate and harmonize disparate information from different applications. Hence, the requirements are:

- Reduce management cost by lowering the number of skilled resources for managing heterogeneous pieces of information.
- Reduction of side effects when implementing information models that use and integrate heterogeneous pieces of information.

2.4.4 *Cloud Computing Challenges and Trends*

As premises in cloud computing, the reduction of payment cost for services and the revenue benefit for investment in proprietary infrastructure are the key factors to believe cloud is the solution to many of the under usage problems and waste in technology the IT sector is facing up. Particular interest for cloud computing services and the use of virtual infrastructure supporting such services is also the result of business model which cloud computing offers, where bigger revenue and more efficient exploitation are envisaged [IBM08]. Likewise, there exist a particular interest from the industry sector, where most of the implementations are taking place, for developing more management tools and solutions in the cloud, and in this way the pioneers are offered full control of the services and the cloud infrastructures. On the other hand, far away from revenue benefits, academic communities point towards finding solutions for more powerful in terms of computing processing and at the same time more efficient to reduce enormous headaches when different technologies need to be interactive to exchange a minimum part of information. Thus, generally problems on manageability, control of the cloud and many other research challenges are being investigated.

Cloud computing management is a complex task [Rochwerger09], for example clouds must support appropriate levels of tailored service performance to large groups of diverse users. A sector of services, named private clouds, coexist with and are provisioned through a bigger public cloud, where the services associated to those private clouds are accessed through (virtualized) wide area networks. In this scenario, management systems are essential for the provisioning and access of resources, and where such systems must be able to address fundamental issues related to scalability and reliability issues which are inherent when integrating diverse cloud computing systems.

2.4.4.1 **Monitoring in the Cloud**

Monitoring in a cloud is essential for automatic or autonomous adaptation to current load, as well as to provide feedback on SLA fulfilment. Scalability and security are essential for cloud monitoring. Without solving the problems of scalability and security, tools and technologies are almost impossible to be deployed in a cloud environment. Moreover, cloud monitoring will also require application-level information to be monitored in addition to the system usage information which current tools can provide [Shao10].

There are several related works in this area. OpenDS [OPENDS] introduces system monitoring for distributed environments and mainly focuses on fault tolerant aspects; NWS [Wolski99] also provides a distributed framework for monitoring and has the ability of forecasting performance changes; When compared to DSMon and NWS, another system resources monitoring tool called DRmonitoring [Domingues03] requires less resources to run and supports multiple platforms (Linux and Windows).

However, up to date of the publication of this book, the DRmonitoring tool lacks scalability and fails to address security concerns. From the industrial viewpoint, HP Open View [[HPOPENVIEW](#)] and IBM Tivoli [[IBMTIVOLISIC](#)] have been developed to ease system monitoring and are primarily targeting the enterprise application environment. Although, the commercial products are relatively limited in portability across different operating systems, they are usually highly integrated with vendor-specific applications, subject to new versions released after the edition of this book.

In the cloud environment, heterogeneity is one of the fundamental requirements for monitoring tools. Therefore, the industrial tools are unsuitable for more general purpose monitoring of the cloud. GoogleApp engine [[GOOGLEAPP](#)] and Hyperic [[HYPERIC](#)] both provide monitoring tools for system status such as CPU, memory and processes resource allocations. Such system usage data can be useful for general purpose cloud monitoring, but they may not be sufficient enough for an application-level manager to make appropriate decisions.

To address the limitations of existing tools, a new monitoring tool named runtime correlation engine (RTCE) [[Holub09](#)] has been developed at the UCD Performance Engineering Laboratory jointly with IBM Software Verification Test teams. RTCE takes the important concerns of heterogeneity, high performance, low overhead and need for reasonable scalability. In the near future, RTCE can be essentially improved with greater scalability based on several proposed architectures [[Wang10](#)]. RTCE will be primarily focusing on stream data correlation and provide flexible data results for other system components to consume. The output data produced should be generic and scalable, so that any type of component can easily adopt the content of the data. In the case of changing signatures on the output, the existing applications should still be able to consume the new output data without any code changes.

2.4.4.2 Non-expert Monitoring of Cloud Services

The need for end users to become involved in cloud infrastructure, service monitoring and management is driven by two key features of cloud computing. As the number of virtual resource “instances” and individual features of resources and services continue to grow to provide flexibility, it will become increasingly unrealistic to expect the cloud provider to manage resources and services for end users.

In addition, as users will pay for resources in a very fine-grained manner, end users may want to monitor their own resources so they can decide when to request more/less resources and ensure they are getting an optimal value for the resources they pay for. A model where user management and monitoring preferences and requirements are mapped to the underlying resources and services in a constrained yet extensible manner, thereby giving users to control over the resources they used to pay.

Harmonization of monitoring data requires mechanisms for mapping of the large volume of low-level data produced by resource-level and service-level monitoring

systems into semantically meaningful information and knowledge that may be discretely consumed by non-experts. Resource-level and service-level experts are generally very familiar with how the constituent parts of the system can be managed, and are particularly aware of the end-to-end operating constraints of those constituent parts. Encoding this expertise in a manner that can be utilized by other stakeholders would enable stakeholders of other systems to monitor other parts of the system and relate their operation to their own system. It also enables common knowledge to be shared and reused.

Supporting personalized visualization of harmonized monitoring data for non-expert end users is viewed as a key to include non-technical “prosumers” in the end-to-end service quality of experience-based control loop [Hampson07]. The relevance of particular monitoring information is dependent on the user-level, application-level, and possibly environmental context, and hence must be properly contextualized [Novak07] in a way that the data consumer appreciates its relevance.

This has led to the popularity of personalized “dashboard” type monitoring applications for visualizing key performance indicators of managed systems [Palpanas07]. A key factor of this approach is to support cloud computing customers in their strategic decisions to invest in more targeted resources, and satisfy themselves that they are getting sufficient value from their investment in cloud computing.

2.4.4.3 Federation–Cloud Interconnection

Federation in the cloud would imply a requirement where user’s applications or services shall still be able to execute across a federation of resources stemming from different cloud providers. It also refers to the ability for different cloud providers to scale their service offerings and to share capabilities to combine efforts and provide a better quality of service for their customers. While the technological aspects required to support cloud federation is an ongoing research domain, there has been little work to support the holistic end-to-end monitoring and management of federated cloud services and resources. This approach requires users and multiple providers to both delegate, share and consume each other resources in a peer-to-peer manner in a secure, managed, monitored and auditable fashion, with a particular focus on interoperability between management and resource description approaches.

Federation in the cloud has the potential to enable new services across a growing range of communication and computing infrastructures to be defined by different providers (or other administrative entities), such that the result enables a transparent service to be available to the end user. Here, “transparent” means that the end user is not aware that the service he or she is using is actually provided by multiple providers. Hence, an important connotation for Federation is that services that a Federation provides should appear to be produced by a single entity. However, this raises two orthogonal challenges: how to support transparent and seamless

interworking/sharing of resources and services for users, and how support operators to securely monitor, manage and share each others' heterogeneous resources to achieve this.

Federation represents an approach for a solution supporting the increasingly important requirement to orchestrate multiple vendors, operators and end-user interactions [Bakker99], [Serrano10], and it is clear in the applicability of this concept in the cloud computing area.

Cloud computing offers an end-user perspective where the use of one or any other infrastructure is transparent, in the best case the infrastructure is ignored by the cloud user [Allee03]. However, from the cloud operator perspective, there are heterogeneous shared network devices as part of diverse infrastructures that must be self-coordinated for offering distributed management or alternatively centrally managed in order to provide the services for which they have been configured. Furthermore, there must be support to facilitate composition of new services which requires a total overview of available resources [Kobiellus06].

In such a federated system, the number of conflicts or problems that may arise when using diverse information referring to the same service or individuals with the objective of providing an end-to-end service across federated resources must be analyzed by methodologies that can detect conflicts. In this sense, semantic annotation and semantic interoperability tools appears as tentative approach solution and that currently is being investigated.

2.4.4.4 Elasticity—Management of the Cloud

The need to control multiple computers running applications and likewise the interaction of multiple service providers supporting a common service exacerbates the challenge of finding management alternatives for orchestrating between the different cloud-based systems and services. Even though having full control of the management operations when a service is being executed is necessary, distributing this decision control is still an open issue. In cloud computing, a management system supporting such complex management operations must address the complex problem of coordinating multiple running applications' management operations, while prioritizing tasks for service interoperability between different cloud systems.

An emerging alternative to solve cloud computing decision control, from a management perspective, is the use of formal languages as a tool for information exchange between the diverse data and information systems participating in cloud service provisioning. These formal languages rely on an inference plane [Strassner07b], [Serrano09]. By using semantic decision support and enriched monitoring information management, decision support is enabled and facilitated. As a result of using semantics, a more complete control of service management operations can be offered, hence a more integrated management, which responds to business objectives. This semantically enabled decision support gives better control in the management of resources, devices, networks, systems and services,

thereby promoting the management of the cloud with formal information models [Blumenthal01].

This section addressed the need to manage the cloud when policies are being used as the mechanism to represent and contain description logic (DL) to operate operational rules. For example, the SWRL language [Bijan06], [Mei06] can be used to formalize a policy language to build up a collection of model representations with the necessary semantic richness and formalisms to represent and integrate the heterogeneous information present in cloud management operations. This approach relies on the fact that high level infrastructure representations do not use resources when they are not being required to support or deploy services [Neiger06], [VMWARE]. Thus, with high-level instructions, the cloud infrastructure can be managed in a more dynamic and optimal way.

2.4.4.5 Support for Configuration and Re-configuration of the Cloud

Several cloud usage patterns can be identified based on bandwidth, storage, and server instances over time [Barr10]. Constant usage over time is typical for internal applications with small variations in usage. Cyclic internal loads are typical for batch and data processing of internal data. Highly predictable cyclic external loads are characteristic of Web servers such as news, sports, whereas spiked external loads are seen on Web pages with suddenly popular content (cf. “slashdotted”). Spiked internal loads are characteristic of internal one-time data processing and analysis, while steady growth over time is seen on startup Web pages.

The cloud paradigm enables applications to scale-up and scale-down on demand, and to more easily adapt to the usage patterns as outlined above. Depending on a number or type of requests, the application can change its configuration to satisfy given service criteria and at the same time optimize resource utilization and reduce the costs. Similarly clients—which can run on a cloud as well—can re-configure themselves based on application availability and service levels required. On-demand scalability and scalability prediction of a service by computing a performance model of the architecture as a composition of performance models of individual components are also features to be considered when designing cloud solution. Exact component’s performance modelling is very difficult to achieve since it depends on a various variables such as available memory, CPU, system bus speed and caches.

2.5 Conclusions

In this chapter...

The requirements of information and pervasive service, based on both NGN demands on context information and the demands of context-awareness according to a set of pervasive service requirements, have been studied and discussed.

The use of ontologies as the formal mechanism to integrate context information for managing service operations is the most adequate. Likewise ontologies is adequate for other management-related activities as policy composition, resources management definition and control and service application representation. In summary the most suitable alternative for supporting interoperability in cross-layer systems. The use of object-oriented models and the policy-based paradigm, both with ontology-based representation, act as a premise in this book. The combination of technologies seems to be a feasible alternative for meeting the convergence requirements of information systems making use of services in NGNs at extensively in cloud environments.

The discussed approaches for ontology-based modelling of context information in pervasive applications cover the information requirements for modelling and structuring of the information in a formal, extensible and efficient manner to collect, distribute and store information. However, current approaches ignore the importance that context information has when it is integrated with management operations. Therefore, a valuable contribution of this chapter is the easy understanding about capabilities and use of information to define and represent management operations.

The main benefit of using policies is to manage the complexity of the management of resources and services in an extensible, scalable and flexible manner. The simplification, which is obtained by providing higher-level abstractions as a result of using ontologies, is also an advantage. Policies are created to help automate the whole system, allowing the adaptation of the services offered according to changing context. This is a crucial goal of pervasive computing systems. It is important to note that this is a unique proposal in this area.

Cloud computing challenges and trends has been studied and discussed in the framework of infrastructures to support cloud services research efforts to provide a clear and valuable state-of-the-art survey, and analysis of cloud trends and challenges has been introduced and discussed. Thus, the use of knowledge platforms that represent and integrate information in various management operations and potentially cloud environments is perfectly justified.



<http://www.springer.com/978-1-4614-2235-8>

Applied Ontology Engineering in Cloud Services,
Networks and Management Systems

Serrano, M.

2012, XXXIV, 190 p., Hardcover

ISBN: 978-1-4614-2235-8