Deliverable 6.2

Part B: Distributed test bed specifications.

Status and Version:	Final	
Date of issue:	15.01.2007	
Distribution:	Public	
Author(s):	Name	Partner
	A. Manzalini, A. Mannella, R. Alfano	TI
	R.Lent (Editor), A. Di Ferdinando	ICL
	Matthias Baumgarten	UU
	T. Al-Bagikni, P. Deussen, E. Höfig	Fokus
	R. Cascella, M. Brunato, R. Battiti	UNITN
Checked by:		

Table of Contents

1 Introduction

The key objective of CASCADAS is the creation of an autonomic toolkit based on distributed self-similar components and characterised by self-* features, such as self-configuration, self-optimisation, self-healing and self-protection. The autonomic toolkit is aimed at establishing a foundation for a next generation service layer, whose purpose is to facilitate the creation, execution and provisioning of situation-aware and dynamically adaptable communication services. Therefore, the CASCADAS project addresses the design of an open service environment, which relies on ACEs (Autonomic Communication Elements) as basic building blocks for the construction of services. As a result, autonomic services will emerge from the composition of ACEs with low development and operational effort.

The next phases of the project aim at prototyping an autonomic toolkit that will offer ACE functionality. The toolkit will be used for the creation of selected application scenarios. As part of WP6 activities, a number of application scenarios have been identified and recognised as particularly useful to demonstrate key research ideas. These scenarios cover a wide range of applications and they will be developed and deployed in a test-bed setting as proof-of-concepts of the project vision of an Open Autonomic Service Framework.

In a more general context, test-beds can serve in two roles: 1) as an evaluation tool on which CASCADAS researchers can quantify techniques against objectives or benchmark different solutions, and 2) as a demonstration platform to promote CASCADAS technology. Three principal test-beds will be available to CASCADAS researchers for their studies. These test-beds are physically located at the premises of CASCADAS' partners ICL, FOKUS and UNITN. It is anticipated that independent studies will be carried out on these test-beds initially. However, there are plans of interconnection at a later phase of the project to conduct larger-scale demonstrations and evaluations. In addition, a test-bed located at UNIMORE's premises will serve to demonstrate specific situation-aware algorithms.

This document describes the foundations of the CASCADAS test-bed design, development and deployment, by addressing the main requirements and obligatory parameters for usability and flexibility. It also discusses specific use cases that will be developed and deployed, and technical specifications and test-bed interconnection.

1.1 Purpose and Scope

This document constitutes Part B "Distributed test bed specifications Document" of Deliverable 6.2. Specifically, it covers a technical description of the CASCADAS' test-beds and selected applications that will be developed and deployed to demonstrate key research results of the project.

1.2 Reference Material

[CASCADAS1] Annex 1 – Description of Work

[CASCADAS2] Minutes of the Bruxelles WP6 4th May 2006

[CASCADAS3] Minutes of the PhC between WP Leaders 21st April 2006

[CASCADAS4] Minutes of the WP 6 PhC 21st June 2006

[CASCADAS5] Minutes of the Belin WP6 18th September 2006

[AdBa66] Adair R., Bayles R. U., Comeau L. W. and Creasy R. J., *A Virtual Machine System for the 360/40*, IBM Cambridge Scientific Center report 36, Massachusetts, May 1966.

[Be05] Fabrice Bellard, "QEMU, a Fast and Portable Dynamic Translator," in Proceedings of the 2005 USENIX Annual Technical Conference, Anaheim, CA, USA, 2005, pp. 41–46.

[BrDr03] Paul Braham, Boris Dragovic, Keir Fraser, Steven Hands, Tim Harris, Alex HO, Rolf Neugebauer, Ian Pratt, and Andrew Warfield, *Xen and the Art of Virtualisation*, in Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles, Bolton Landing, NY, USA, 2003, pp. 164–177.

[BuBa03a] Mauro Brunato, Roberto Battiti. PILGRIM: A Location Broker and Mobility-Aware Recommendation System. Proceedings of PerCom2003, Fort Worth (TX), USA, march 2003.

[BuBa03b] Mauro Brunato, Roberto Battiti. A Location-Dependent Recommender System for the Web. Proceedings of the MobEA Workshop, Budapest (Hungary), may 20 2003.

[Cr81] Creasy R.J., *The Origin of the VM/370 Time-Sharing System*, IBM Journal of Research and Developments 25 (Sep 1981), pp. 5.

[Di06] Jeff Dick, *User Mode Linux*, Prentice Hall, ISBN-13 97-801-31865051, 2006.

[FeSc99] Niels Ferguson, Bruce Schneier, *A Cryptographic Evaluation of IPSec*, unpublished manuscript from BT counterpane, available online: http://www.macfergus.com/pub/IPsec.html, February 1999

[Intel01] Intel Corp., *Architecture Software Developer's Manual, volume 2: Instruction Set Reference Manual*, Intel®, Santa Clara, Order Number 243191.

[Intel02] Intel Corp., *Vanderpool Technology for IA-32 Processors (VT-x) Preliminary Specification*, Intel®, Santa Clara, Order Number C97063-001.

[Intel03] Intel Corporation, "Preboot Execution Environment, (PXE) Specification Version 2.1," Intel® 1999.

[KeSe03] Kenneth B. Kent. and Micaela Serra, *Reconfigurable Architecture Requirements for Co-Designed Virtual Machines*, Proceedings of the 17th International Symposium on Parallel and Distributed Processing, IEEE Computer Society, Washington, DC, USA, 2003, pp. 187-189.

[KiSm03] Ho-Seop Kim and James E. Smith, *Dynamic Binary Translation for Accumulator-Oriented Architectures*, Proceedings of the international symposium on Code generation and optimation: feedback-directed and runtime optimisation, IEEE Computer Society, San Francisco, California, USA, 2003, pp. 25-35.

[Le05] R. Lent. Design of a manet testbed management system. *The Computer Journal*, 49(4):171–179, July 2006.

[LiFl02] Hans Liebig and Thomas Flik, *Rechnerorganisation. Prinzip, Strukturen, Algorithmen*, Springer, Berlin, ISBN 3540000275, 2002.

[MeBo05] Mira Mezini, Christoph Bockisch, Tom Dinkelaker and Michael Haupt, *Lecture: Virtual Machines*, Technische Universität Darmstadt, Darmstadt, 2006.

[NeBr05] Newman M., Braswell C.M. and Wiberg B., eds., *Server Consolidation with VMware ESX-Server*, International Business Machines Corporation, Redpaper, http://www.redbooks.ibm.com/abstracts/redp3939.html, 2005.

[PoGo74] Gerald J. Popek and Robert P. Goldberg, *Formal requirements for virtualisable third generation architecture*s, Communications of the ACM, 17 (1974), pp. 412–421.

[Si06] Amit Singh, *An Introduction to Virtualisation*, Kernel thread, http://www.kernelthread.com/publications/virtualisation/, 2006.

[SmNa05] James E. Smith and Ravi Nair, *Virtual Machines: Versatile Platforms For Systems And Processes*, Morgan Kaufmann Publisher, ISBN 1-55860-910-5, 2005.

[Tr05] Win Treese, *Virtualisation virtually everywhere*, ACM Press, netWorker, 9 (June 2005).

[VmWare06] VMware Inc., *Accelerate Software Development, Testing, and Deployment*, VMware White Paper http://www.vmware.com/pdf/dev_test.pdf, 2006.

[WoHa05] Chris Wolf and E. M. Halter, Virtualisation: From the Desktop to the Enterprise: Apress, ISBN 1-59059-495-9, 2005.

[YuJie06] Chen Yu, Ren Jie, Zhu Hui and Shi Yuan Chun, *Dynamic Binary Translation and Optimisation in a Whole-System Emulator – SkyEye*, Proceedings of the 2006 international Conference Workshops on Parallel Processing, ICPPW. IEEE Computer Society, Washington, DC, USA, August 14 – 18, 2006, pp. 327-336.

1.3 Document History

Version	Date	Authors	Comment
0.1	15/06/2006	A. Manzalini (TI)	Initial documents
0.2	13/09/2006	A. Manzalini (TI)	Revised Draft after WP6 PhC and Ricardo's contribution
0.3	03/10/2006	A. Manzalini (TI)	Revised Draft after WP6 meeting in Berlin (19/09/2006)
0.3.1	23/11/2006	A. Di Ferdinando (ICL)	Amended description of the Distributed Auctions scenario.
0.5	11/12/2006	R. Lent (ICL)	General edition, UNITN, UNIMORE, TI contributions
0.6	14/12/2006	E. Hoefig (Fokus)	Added Fokus contribution
0.7	27/12/2006	R. Lent (ICL)	General edition

1.4 Document overview

The document is structured as follows. Section 2 presents an overview of the future CASCADAS test-bed. In section 3, use cases of selected application scenarios to be developed and deployed in the test-bed environment are described. Section 4 discusses the test-bed requirements and design. A roadmap of next WP6 activities related to the establishment of the test-bed is discussed in section 5. Finally, section 6 ends the document with a description of alternatives for test-bed interconnection.

2 Test-bed Overview

A key milestone of the CASCADAS project is the design and deployment of a distributed test-bed that will be used to evaluate, help improve and demonstrate a prototype of an autonomic toolkit based on distributed self-similar components. A number of representative applications have been selected to demonstrate in the test-bed environment, a real-time execution and provision of situation-aware and dynamically adaptable communication and content services. The results will validate the project vision of an Open Autonomic Service Framework, which aims at developing a next generation service layer. Figure 1 outlines a model of the CASCADAS test-bed, which has been broken down for discussion purposes into four conceptual layers: application, service, network and users.

The user layer covers person-to-person and person-to-environment types of communication services, which might require situation-awareness for the provision of specific services, for example, in the provision of behavioural personal advertisement services. Users might offer knowledge to enable enhanced functionally, such as advice about potential problems in certain areas of the network based on their personal experience. In a wider scope, the application layer covers general purpose and dynamically adaptable services to users.

In the model, the network layer provides the means for moving information across users and applications, and implements continuous monitoring of status. Network status information, such as the available bandwidth of communication links, might become available to the service layer to enable the provision of autonomic solutions, such as fault recovery, load balancing, etc.

The service layer will be offered by the autonomic toolkit, which will provide the means for the creation and execution of services and the provision of the Open Autonomic Service Framework. The service layer forms a dynamically reconfigurable overlay network relying on an autonomic composition of distributed resources and decentralised functionalities, taking advantage of external information offered both by the application and network layers.

The design of the CASCADAS test-bed and experiments with selected application scenarios will be defined around this model, to demonstrate in a realistic environment the advantages of the Open Autonomic Service Framework in the provision of situation-aware and autonomic services.

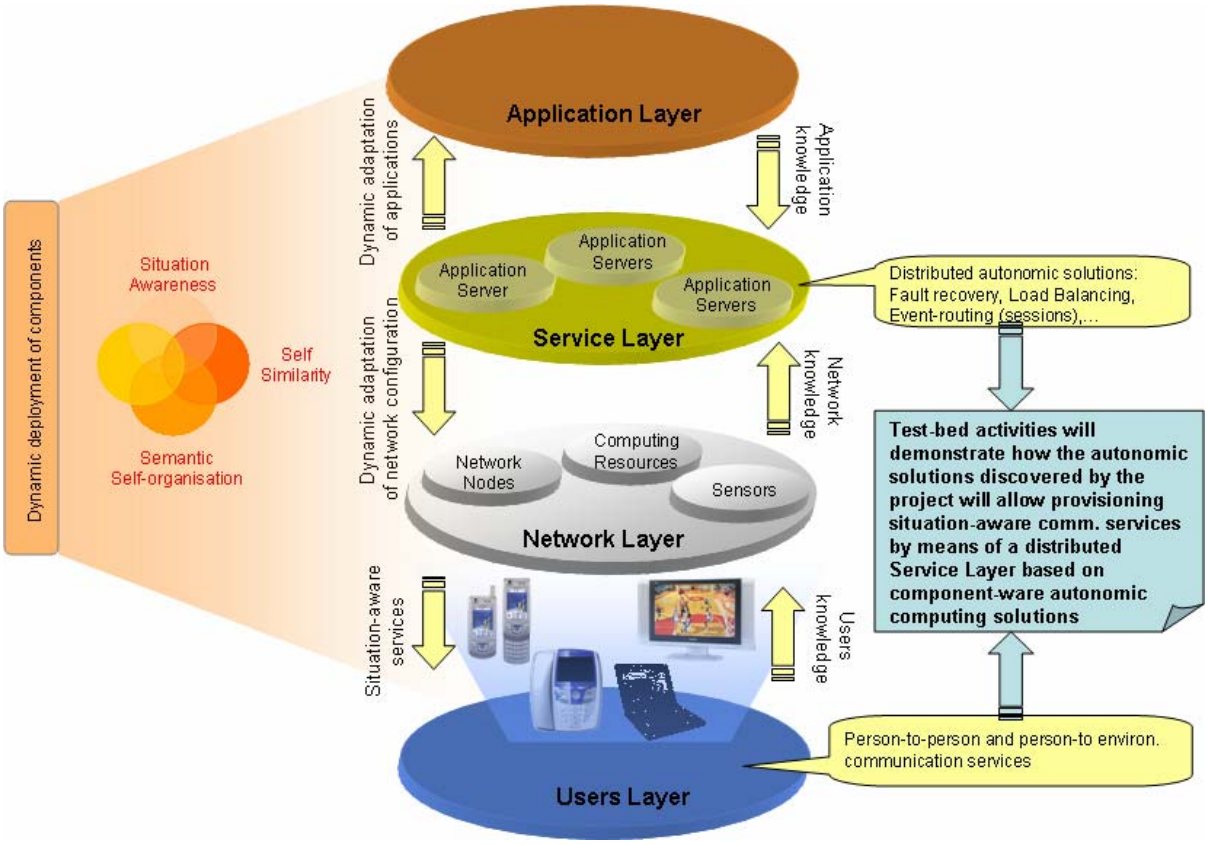

Figure 1 - CASCADAS test-bed

3 Use-cases to validate the CASCADAS framework

This section describes a number of use cases of selected application scenarios to illustrate some of the expected results of WP6 during the next phase of the project.

3.1 Service for Pervasive applications

Within such general scenario, we foresee to demonstrate two different classes of services: (i) smart environments services for supporting independent living; (ii) behavioral pervasive content sharing services. In practice, such kinds of services, though serving different purposes, all represents instances of either people-to-environment coordination or of people-to-people coordination. Let us now recall the key characteristics we envision for such general scenario.

First, the scenario considers an environment, which is densely enriched with sensorial and computational capabilities. In particular, sensor networks as well as RFID tags are embedded in the environment, and can act both as sources of environmental data as well as a sort of environmental computing infrastructures. Sensors can interact with each other in an ad-hoc way, with some users nearby, and can possibly (not necessarily) be connected to some "sink-server" that can be used to collect data. RFID tags can be accessed by nearby users and devices to read and/or store data.

Second, humans populate the scenario, live in it and interact with each and with each other. It is expected that users carry with them some kind of mobile devices (e.g., smart

phones and/or PDAs). Via such portable devices, users can be given access to the information produced by nearby sensors and tags, can possibly interact with each other in an ad-hoc way (e.g., via bluetooh or WiFi), and can possibly access to the Internet via some wireless connection.

As discussed in more detail in D6.1 part A, the above scenario can soon become reality. For example, a big exhibition center may already have the resources to invest in making it a reality, so as to make it possible to deliver, as an additional value of the exhibition, a number of innovative services to guide people in better enjoying the exhibition, to socialize with each other, and to support visitors with limited abilities.

Of course, since it is impossible to actually deploy a test-bed of a pervasive exhibition, the demonstration will focus on a small scale pervasive computing test-bed, where some pervasive devices such as sensors and RFID tags within an locally-confined environment (e.g., a hall) are exploited to enforce the exploitation of situation-aware services by a number of mobile users. Overall, the demonstration could be organised as follows (see Figure 2):

- The general architecture considers users (circle 1 in Figure 2) equipped with portable computing devices (i.e., laptops or PDAs), as the clients of pervasive services.
- Such portable devices can integrate localisation devices (i.e., GPS – circle 6 in Figure 2) and devices to acquire information from the physical world (i.e., RFID readers and sensors, circles 4 and 5 respectively in Figure 2).
- Also users have supposed to have means to connect to the Internet (i.e., WiFi and/or UMTS connections, circle 7 in Figure 2) and/or to interact with each other in an ad-hoc way (circle 8 in Figure 2).
- Services, implemented via proper ACEs, execute on the users' portable devices (circle 2 in Figure 2) and, in the case of distributed services made up of a number of interacting ACEs, can be implemented with the support of ACEs executing remotely (circle 3 in Figure 2).
- Similarly, contextual information, in the form of proper knowledge networks, can be acquired either remotely (via ad-hoc interactions with other devices or from the Web) or directly from the environment (as it can be provided by accessing RFID and sensors around).

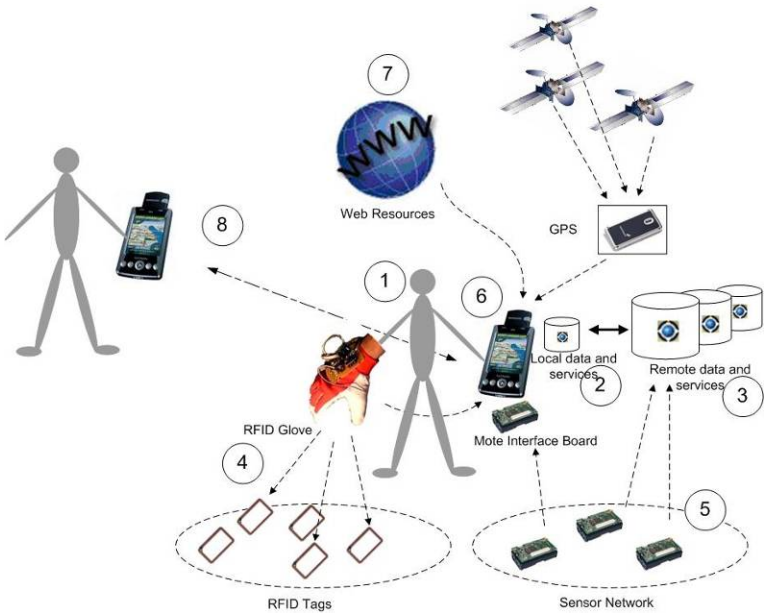

Figure 2. Demonstration Test-bed for Pervasive Services

Based on this architecture, we will be able to demonstrate in practice how the CASCADAS concepts can be effectively used for the design and deployment of effective pervasive services. In particular, with reference to the services described in D6.1, it will be possible to use this architecture in order to demonstrate the following services:

- **Behavioural pervasive advertisement**, which is a service to ensure that advertisers reach the target audience in a more effective way. In the envisioned scenario, this may results in specific advertisement being directed in a personalised way to individual PDAs, and/or in wall displays at the exhibition to show those ads that, at a given moment, would better fit the surrounding audience.
- **The Living Diary**, which is a personal user-centric applications, which aimed at exploiting the pervasive devices embedded in an environment to produce a sort of digital self-composing diary. This is a very useful services to enable people to better interact with the surrounding physical world, and can also be of use as a cognitive reinforcement to people with cognitive problems.

Strictly related to the above Living Diary Example, we will be able to demonstrate some simple services for supporting independent living (see also D6.1), such as "**Wherareyou**", "**GoTo**", which helps people to navigate in a physical environment and other similar services introduced in D6.1.

In a first release, the above services will be realised in the test-bed infrastructure via means of ACEs that will be able to dynamically aggregate and interact with each other in a reliable and autonomic way (thus demonstrating some key WP1 features), that will become situation-aware by accessing knowledge networks (thus demonstrating some key WP5 features), and that will exploit some basic mechanism of self-monitoring and self-preservation (thus demonstrating some WP2 features). In a second release, we expect to be able to integrated advanced security solutions (to demonstrate WP4 features) and advanced self-organisation features (to put WP3 algorithms at work).

The demonstration of more advanced and realistic services to support independent living is worth a separate discussion. In fact, from a technical perspective, addressing the problem

of realising realistic pervasive services to support independent living is largely more complex for a number of reasons. In the first instance, the service must be delivered to available stakeholders at a number of levels of granularity. For example, provision of service with those in close proximity perhaps in the same building or street, or those in the same town or alternatively those who may be simply 'available' to offer support but may be located within a larger geographical area. A secondary issue for consideration is are the people or services within proximity able to provide support at all and if so at what level? For example a person who requires some advice as to why they have left the house may only be able to receive support in this instance by a family member who has access to the person's daily agenda. On the other hand, if the stakeholders' prosthetic raises an 'emergency suggestion for intervention' message then the appropriate stakeholder who can be contacted should be made aware of the situation.

If we assume in the first instance that communication between the person's cognitive prosthetic and the stakeholder's prosthetic can be established the problem becomes one of complex information management within a networked environment addressing the dynamic sensing of very detailed information of the person and stakeholders. This calls for general system requirements that go beyond the simple test-bed of Figure 2. For instance, we may require advanced sensing techniques to acquire knowledge from motion sensors, pressure sensors, perhaps some vital signs like Heart Rate or blood pressure. Also, this may require advanced computational system to perform complex knowledge management tasks. Thus, at present we believe that such realistic scenarios can only be demonstrated in a virtual, that is simulated, environment, although realistic case scenarios for persons that require individual support in their living environment and beyond could be used. Additional recourses would be necessary to implement a demonstrator that incorporates a real world test environment

3.1.1 Use-case: Pervasive Services supporting independent Living

3.1.1.1 General System and Service Requirements

From a technical perspective addressing the problem of realising pervasive services to support independent living is largely complex for a number of reasons. In the first instance the service must be delivered to available stakeholders at a number of levels of granularity. For example, provision of service with those in close proximity perhaps in the same building or street, or those in the same town or alternatively those who may be simply 'available' to offer support but may be located within a larger geographical area. A secondary issue for consideration is are the people or services within proximity able to provide support at all and if so at what level? For example a person who requires some advice as to why they have left the house may only be able to receive support in this instance by a family member who has access to the person's daily agenda. On the other hand, if the stakeholders' prosthetic raises an 'emergency suggestion for intervention' message then the appropriate stakeholder who can be contacted should be made aware of the situation.

If we assume in the first instance that communication between the person's cognitive prosthetic and the stakeholder's prosthetic can be established the problem becomes one of information management within a networked environment addressing the dynamic positioning of the person and stakeholders. This suggests that the requirements of the technical service should provide for the following:

- Situation awareness – to identify the positioning of the person concerned and relevant stakeholders
- Self organising – to identify the closest stakeholder in instances of alarm
- Autonomy – the ability of the service to dynamically self adapt
- Knowledge management – the ability to infer from the knowledge within the network

Given we wish to exploit such services it may be possible to use a dedicated laboratory environment with selected domotic devices in order to provide suitable test environment on a smaller scale. The general system requirements that have to be realised before such services could be deployed efficiently could be summarised as follows:

- The provision of smart world infrastructures: Reaching from single sensors over localised smart environments to global and openly accessible smart infrastructures that are necessary to achieve situation awareness at all relevant levels of granularity.
- Unique identification and flexible communication methods of and among sensors, devices, people etc. in order to enable self-organisation among relevant entities and to implement and coordinate support activities of and among individual entities.
- A number of distributed computational resources to enable advanced knowledge management techniques and to access dedicated services

For example, we may have knowledge from motion sensors, pressure sensors, perhaps some vital signs like Heart Rate or blood pressure. These could all be the core elements which would provide relevant knowledge for the knowledge network (WP5) and be used in some means as an aggregate to actually facilitate knowledge support for pervasive services in this domain. This may be appealing due to the fact that two people's homes are similar while their activities will be largely different hence it will be possible to demonstrate this adaptability and perhaps the ad-hoc nature of the service by using a number of different people within the same test environment. The supervision element (WP2) could be introduced to show how the environment could adapt to the person's behaviour. For example, if there is significant wandering within a room or between rooms the system or service should react in some way.

3.1.1.2 What will be demonstrated?

Based on the example scenario outlined in Deliverable 6.1 and assuming that a large scale smart world infrastructure or a more local smart environment is available, either real or simulated, the following demonstration activities are envisioned.

- **People to Environment (localised) Interaction:** Consider a localised smart environment as e.g. as depicted in Figure 1, which illustrates a home environment. The flexible and dynamic interaction between such an environment and its inhabitants, and vice versa, offers a wide range for possible pervasive services. Such services could be applied to:
    - Raise alarms in the case of dangerous situations.
    - React to the specifics of individual inhabitants.
    - Organise daily life activities.
    - Etc.

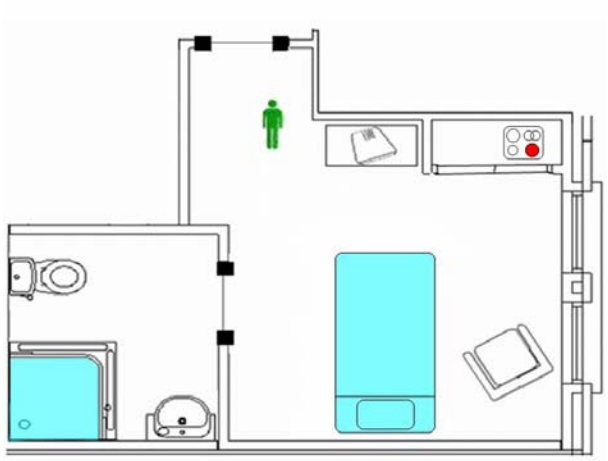

Figure 1: Smart Home Environment

- **People to Environment (global) Interaction:** For this, consider a more global oriented smart world infrastructure such as depicted in Figure 2. Such an environment is by far more dynamic, that is the type and amount of available contextual information is much more diverse and larger, respectively. Furthermore, the types of services to be applied may differ for different locations. For instance, the same type of service in two different towns may produce a different result. Therefore, the type of possible services, for such an environment, is virtually unlimited. Nevertheless, some of the most interesting once may stem from the areas of cognitive and spatial reinforcement where the following example services represent suitable demonstration tasks:
    - Cognitive reinforcement (helping people to remember)
        - GoTo(Location(x))
        - This may be considered as a form of person-to-environment interaction.
        - This may be considered as a form of person-to-person interaction
    - Spatial reinforcement (in order to improve and extend a person' mobility)
        - AlertCare(IfPersonLost(Client(x)))
        - WhereAmI?

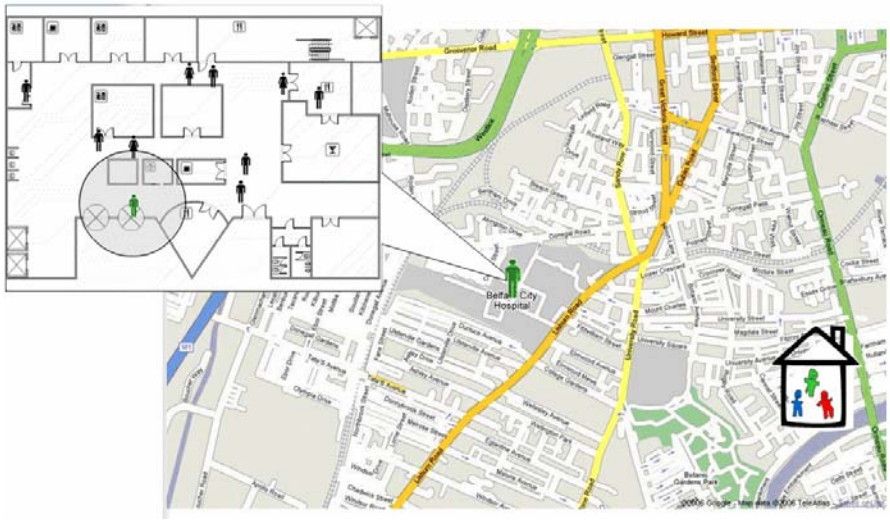

Figure 2: Smart World Infrastructure

- **People to People Interaction:** The interaction between people or applications, in this case, represents another area where dedicated pervasive services can be applied to. In this case specific interactions may be invoked; configured or individual personal interest may be shared to achieve separate goals. As depicted in Figure 3, the identification of individual interaction entities (people or applications) and the negations of available and needed services represents a distinct objective of the project and thus forms the basis for possible services such as:
    - Social reinforcement (aiming to help people to maintain social contact inside and beyond their own family)
        - Call(Person(x))
        - WhereIsPerson(y)
        - WhoIsClosestPerson(x)
        - This may be considered as a form of person-to-person interaction
    - Functional reinforcement (to help perform daily life activities)
        - EatLunchAt(Time(x), Restaurant(y))
        - A form of person-to-environment interaction to provide direct assistance.

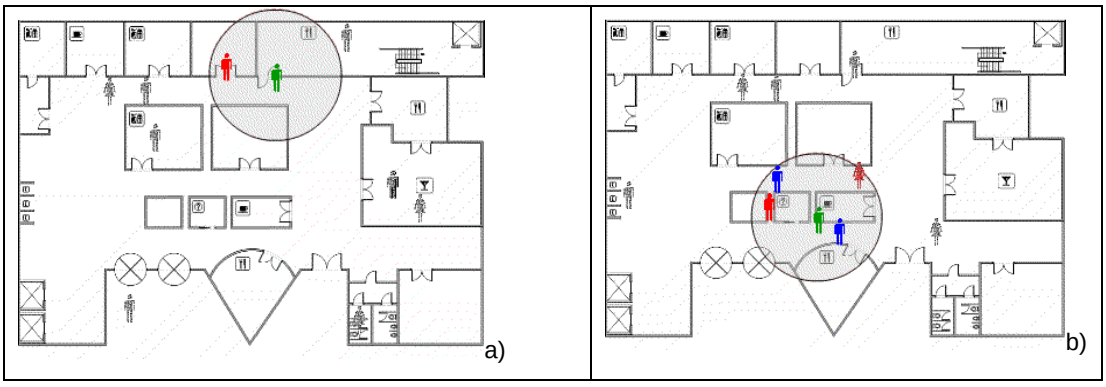

Figure 3: People-to-People Interaction (a) Person can obtain information from a single stakeholder (b) Person can obtain information from a hoist of stakeholders.

The actual focus and scale of the demonstration activities will, obviously, strongly depend on the contribution of each work package towards the overall framework but also towards specialised configurations thereof, which are required to address the specifics of this scenario.

At present we believe that this scenario can only be demonstrated in a virtual, that is simulated, environment, although realistic case scenarios for persons that require individual support in their living environment and beyond could be used. Additional recourses would be necessary to implement a demonstrator that incorporates a real world test environment.

3.1.2 Use-case: Scenario Behavioral Personal Advertisement

Behavioral Pervasive Advertisement (BPA) applies pervasive and autonomic computing to the nowadays emerging advertisement technique called *behavioral advertisement* (or behavioral targeting) aiming at advertising targeted audiences in a very effective way.

In principle, behavioral targeting should allows marketers to better grasp customers' needs and interests, mainly by tracking and monitoring consumer behaviors on the web. The use-case proposes to extend such advertising techniques to a general person-to-person, person-to-environment communication context where Users' interests and needs (according to their authorisation) can be grasped.

Moreover, exploiting the pervasive nature of CASCADAS based applications, BPA may provide customised contents and advertisement, not only during web navigation, but different channels may be personalised to the single user or to groups of users (e.g. digital screens in the road may autonomously provide advertisement customised to the profiles of the user moving in the zone of the screen).

Behavioral Pervasive Advertisement is considered and excellent use-case to demonstrate the CASCADAS autonomic component-ware framework to the key role played by autonomic communications.

The service provider should be able to respond to users needs as they arise, by implementing *a pull-based* VAS offer, in which customers can access a set of functionality in multiple ways, driven by their context and interests. Such functionality should be dynamically composed and made available by the autonomic features.

To enable that vision, it is necessary that the service development and delivery infrastructure available to Telecommunications operators provide means:

- To maintain a large and ever growing portfolio of basic service building blocks;
- To allow the available building blocks to rapidly self-organise them exploiting context information;
- To get user habits and lifestyle from the communication services provided by the operator.

3.1.2.1 What will be demonstrated

The Behavioral Pervasive Advertisement system will heavily depend on the combination of quasi-static user profiles with dynamic context data describing the user's current status in terms of physical position, current activity and other relevant information. Such combination of data will enable the system to provide time- and position-sensible information to customers, therefore improving the effectiveness of the displayed information.

User profile information can be maintained by means of a loop where users provide feedback information to the system. Such feedback information can be either explicit (vote) or implicit (e.g. time of permanence in front of the display), and can be continuously merged into the system, together with information describing the actual context in which the advertisement information was delivered [1,2]. The importance of context information can be understood by many examples: for instance, the same restaurant ad can be very appealing at noon and pass unnoticed at mid afternoon; in many cases, this relationship can be predetermined and hard-wired in the system. However, inspection of correlations between context data and user feedback can unveil previously unexpected relationships.

Besides the time of day, relevant context data might include position, crowd density, and presence of similar people, even weather conditions. An autonomic network can provide aggregate context information by fusing data from a number of sensor nodes; detection of similar people in the same place can be operated by appropriate ACEs that implement learning/optimisation algorithms to identify clusters and cliques in relationship graphs.

By default, the pervasive advertisement system will operate as a display that alternates among all possible ads; once the presence of any user is sensed in a proximity area, the system will switch to its autonomic behavior. The demonstration of the feasibility of the scenario can be conducted by providing the system with detailed user profiles and with all the context information required to trigger the display of the appropriate advertisement. If many users dynamically interact with the system at the same time, the screen should determine the advertisement that best matches all the different profiles by employing the clustering functionality of the optimisation-capable ACEs.

The adaptive features of the systems will be tested by modifying user profiles, context information and/or specific features of the system. For instance a variation of the location of the user or the information on the mobility of the user will trigger different advertisements.

From the security and survivability viewpoint, we can expect this system to react to false information introduced by malicious parts of the system in order to increase their utility beyond a reasonable point. For instance, a single mobile ACE could decide to impersonate many similarly-profiled nodes in order to trigger the visualisation of a particular ad. Security and self-management properties of the system can be demonstrated with the application of a reputation system as the system relies on user feedbacks to advertise points of interest or other information. The trustworthiness of the components interacting in the system might exclude from the system malfunctioning components or malicious users who attacks the system to modifying opportunistically the advertisement that should be shown.

3.2 Non-pervasive, communication-intensive applications

3.2.1 Use-case: Autonomic Distributed Auctions

3.2.1.1 Summary and rationale

The scenario models future economies, which we expect to be organised around networked and completely automated transactions between enterprises, and between individuals and enterprises. Such systems are expected to carry a high number of short-lived electronic transactions operating at a high frequency. To succeed, auction participants will need to operate in an opportunistic way.

The support of an autonomic communication network will ensure the delivery of uninterrupted economic services with a defined, yet customisable, *Quality of Service* (QoS) to a large number of users, which can be automated, such as agents, or physical users. Such communications will exploit aspects, typical of autonomic computing, such as situation-awareness and self-adaptation to network conditions (in order to deliver the fastest response time and the best protection against failures) and self-protection techniques (against malicious users). Likewise, self-configuration would allow auctions to virtually move in the network to "position" participants in the most advantageous places and allow them maximise their revenues. Operating at a global scale, the managing complexity of this application would scale up without control unless autonomic support emerges.

3.2.1.2 Users

The scenario will consist of several nodes physically located on different networks, interconnected each other to form a single bigger network. This platform will be populated by:

- *Buyers*: users searching for, and willing to bid for, items of interest.

- *Seller*s (or *auctioneer*s): users owning items that sell them through auction. These users are also said to *start* an auction by advertising the item.
- *Auction Centers* (ACs): represent markets where buyers and sellers meet. Contain representation of (sub-system of) networked auctions, typically visualised as *Auction Web Page*s (AWPs). There contain the list of items currently auctioned along with a description of auction terms and conditions. ACs have an influence area that might be related (or not) to a geographic position, or to a particular type of products or services.

3.2.1.3 Basic course of events

Sellers and bidders will advertise their goods (or services) through ACs by providing a description of the item to be advertised, containing all needed information eventual users interested in participating to the auction need to know. In detail, by means of an advertisement sellers will publicly notify their will to sell an item according to a set of rules (there also provided), while buyers will publicly notify their will to buy an item (eventually under certain auction rules).

Typically, information contained in an advertisement will include initial price for the item, starting and expiration date (for timed auctions) and the model of auction (or a set of rules regulating the auction, if these cannot be arranged into a well known auction model) chosen for the process. ACs will enable some level of control over the auctions, for example, by regulating access control, including qualification and verification of users' credentials.

Buyers will find sellers auctioning goods of interest (and sellers, buyers willing to participate to auctions on goods they want to auction) by consulting ACs on demand. Interested buyers will then contact sellers (and, conversely, sellers will contact buyers) privately in order to proceed with the auctioning process. Updates on auctions being carried out will be instantly reflected in ACs.

To succeed in auctions, participants will act in an opportunistic way: self-configuration will allow virtual movements of participants, that will be able to position themselves in the most advantageous places in the network, in such a way to maximise their chances to succeed in the auction process. This location, which we have named *Virtual Location* (VL), will be determined with respect of certain QoS requirements to fulfill, and will require the support of the autonomic communication network in a way that it will determine the ideal position to migrate to in the network. In addition, autonomic elements will be necessary to facilitate the exchange of *Auction Message*s (AMs). As it can be figured out, there are stringent communication requirements (including migrations) of a high complexity in the process of finding offers and in conducting the rational process of the auctions. As an example, let us consider location in a time-constrained auction: bidders capable of reaching the seller quicker (e.g. closer to the seller in terms of end-to-end delay) will have better chances to snipe the auction than bidders a few milliseconds further away

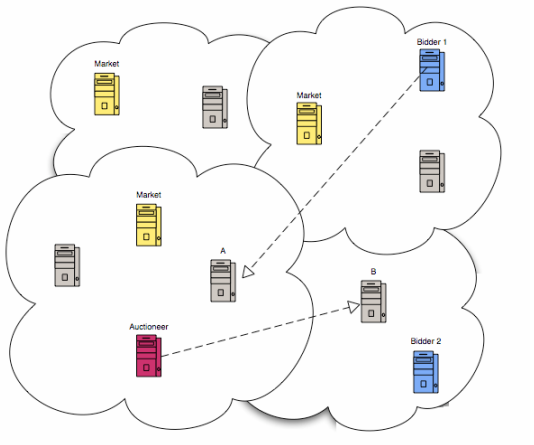

Figure 4: Auction Scenario

from the seller. This creates an undesirable position to certain participants. Another example can be seen in auction participants connecting from networks of different characteristics. In fact, different networks provide different communication service levels, which produce a differential effect (e.g. a wireless user may have higher loss probability). The complexity scales up further if we consider that users will often interact with more than one AC to try to increase their profits.

3.2.1.4 What will be demonstrated

The distributed test bed will consist of a number of nodes geographically distributed, running the Distributed Auction (DA) platform. On each node, users will have the option of choosing manual or automated auction management. In the latter case, the user will specify an overall *utility* that the application will aim at achieve. When automated management is selected, the application will have the freedom to decide participation in auctions while attempting to maximise revenue (utility). As an example of utility, the user might ask the application to manage a company's warehouse by requiring a fast turnover of goods. In the manual case, the user will manage auctions herself by deciding auction participation.

As a second part of the management choice, when the application is managed in an automated way, an *agent* will randomly generate items to auction (as a seller) and join auctions (as a buyer). The agent will reside at application level (i.e. will be an *application-level agent*) and will be in charge of instructing the application to generate random items (along with the correspondent description) to be auctioned based on random auction models. However, in contexts where the application will act as seller the agent will be smart enough to preserve consistency between the nature of the generated item and the rules under which to auction it.

The application will visually display the current activity of the AU (and, eventually of the whole network) by providing detail of each auction the local application is currently involved in with all relevant data such as description of the item under auction, enumeration of other participants (whereas the auction model allows these to be known), current price, bid history and so forth.

After creation, the item will be advertised at one or more ACs. This latter will be reachable by means of connections on specified IP addresses and will essentially contain an AWP. Items in the AWP will be represented by an object containing everything necessary for a hypothetical bidder to participate. The AWP will be linked to a database containing items objects. Buyers' request of specific items will query the database to the extent of obtaining a list of relevant item objects based on a set of characteristics specified by the buyer in the query.

The actual bid will be placed by means of a point-to-point connection with the auctioneer, and this will mark the start of an auction from the perspective of the buyer. Point-to-point communication between auctioneer and bidders will allow all parties involved to interact in the context of the auction, while other point-point communication between auctioneer and the ACs currently containing the advertisement for the item under auction will allow reflecting every change in the item in the involved ACs.

The test-bed is expected to highlight the benefits of autonomic computing in the process of auctioning. In particular, aspects tied to self-configuration and self-mobility will allow the AU to act opportunistically by migrating to other nodes in order to maximise the chances to succeed in the auction. In addition, exploitation of self-aggregation will allow dynamic creation of new, more sophisticated services. For instance, when an AU finds an auction

for an item of interest auctioned under an auction model it cannot correctly interpret, it might aggregate to another AU to the extent of forming a new AU capable of interpreting rules from *all* auction models correctly interpreted by the two original AUs. However, the AU chosen for aggregation might request payment for the actual aggregation to take place. This is an aspect that the AU is expected to handle autonomically, based on the available budget and the overall convenience.

4 Test-bed Design

The design of the CASCADAS test-bed has a life cycle, which consists of four principal phases. These phases will be carried out in a cyclical way, so that the revisiting of phases should help in improving the design of ACEs and the test-bed itself with every cycle.

1. Requirement analysis. Consist in identifying the key needs to be fulfilled both in defining the infrastructure and software development. Most requirements can be recognised from the specific characteristics of the application scenarios to be deployed and the particular needs of the autonomic toolkit.
2. Physical design. Consists in selecting specific technologies and products to materialise the test-bed. Includes the selection of computer hardware, operating systems, communication service providers, etc.
3. Logical design. Includes the integration of the the autonomic toolkit and supplementary tools for experimentation and demonstration purposes of application scenarios. The logical design also includes setting overlay networks and computer virtualisation to increase the available nodes in the network.
4. Evaluation and Optimisation. The final step after constructing the test-bed and deployment the applications is the evaluation and optimisation of the design to identify potential amendments.

4.1 Basic Requirements

The following are various properties that are desired in constructing an effective test-bed for CASCADAS. The requirements have been identified from expected experimentation needs in addition to software development/maintenance and logistic necessities. It is important to notice that these requirements have appeared from the information currently managed across different WP of the project. However, the list is not necessarily exhaustive and new requirements may appear in the future as project research progresses.

- Modularity. Similar to the software construction approach of ACEs, the test-bed should be modular and it should permit an easy exchange of components for easy experimentation. Also, it should be possible to test implementations of diverse capabilities for comparison and validation purposes without disrupting on-going experiments.
- Controlability. The test-bed should permit an easy adjustment of parameters for a variety of applications. It should also allow for introducing perturbations into the system, which are intended to trigger autonomic behaviour.
- Observability. Experiments need to be quantified via data collection both in a centralised and decentralised way. Centralised data collection will be particularly useful in demonstration activities where a user interface will show application

behaviour on-line. In evaluation studies, decentralised collection will be useful in most cases.

- Availability. Researchers should have the possibility of conducting experiments without limitations of geographic location or time.
- Security. Access to the inner details of the CASCADAS test-bed should be restricted with proper authorisation and authentication, in particular, during the initial phases of development.
- Universal access. In later phases of development, certain portions of the test-bed may become public to be a general demonstration tool or "display window" of on-going experiments and key research ideas.
- Uniform tools. A set of common tools should be made available to CASCADAS' researchers across the distributed test-bed. Such tools might include a common development framework, libraries, etc.

4.2 Structure

The CASCADAS test-bed structure consists of two main parts: infrastructure and custom-built software.

The infrastructure part has three principal local test-beds deployed at premises of ICL, FOKUS and UNITN, which will be interconnected to form a larger test-bed. A specialised test-bed located in premises of UNIMORE will be devoted to specific demonstration purposes within the project. Appendix A lists the equipment available at each location.

The custom-built software part consist of the following parts:

- ACE toolkit. An ACE toolkit will be developed and made available as a project-wide milestone and it will be used to support the development of autonomic services. The ACE toolkit will be deployed in the CASCADAS test-bed as a component to be used by the applications.
- Selected applications. Application scenarios are described in detail in Deliverable 6.1-Part A. Among these applications, there are a number of them devoted to the applicability of the autonomic principles for the evolution of the Service Layer. These particular applications, for example, next generation service infrastructures and ubiquitious grid computing for pervasive services, will be used as evolutionary architectural enablers that will assist in the provision of essential ACE functionality and the definition of the Open Autonomic Service Framework.

  Application services will be developed from ACEs along two directions to maximise application coverage. The first direction will focus on autonomic communication aspects, for example, situation and context awareness. The main idea is to be able to evaluate and demonstrate pervasive applications. At least two applications will be developed and deployed in test-beds: smart environments supporting independent living and behavioral pervasive content sharing. The second direction is along supporting communication-intensive applications. We will develop and deploy a distributed auctions application.
- Workload generators. Although interactions with real users are expected in the experimental evaluation and demonstration of the autonomic toolkit and applications, a synthetic source of workload would be required for convenience in most cases to persistently conduct tests for long periods of time.

- Monitors. Dedicated monitors will observe and record events of interest in the systems under study. Such events might be triggered for example, by the application of autonomic properties of the system, which we would like to measure.
- Control. Another piece or pieces of software will control the execution of experiments, e.g., by injecting specific perturbations into the system to enable autonomic properties to emerge.

4.3 Using Virtualisation for Testbed Implementation

To achieve flexibility and extendibility in the CASCADAS testbed, a solution based on virtualisation will be proposed as an approach to create an additional layer allowing for independency in regards to the employed hardware, operating system, and network technology. The basic idea is to construct the testbed from machines that are configured in a very basic and simple manner (hereafter called testbed "nodes"), running a minimal operating system and exposing the capability to start virtual machines that are able to emulate a wide range of existing operating systems and middleware technologies. In this scenario a developer wishing to evaluate, e.g., an ACE design choice would create a virtual machine image on her personal workstation and configure an appropriate runtime environment, system libraries, network setup, etc. for the test. Afterwards the image would be distributed to the testbed nodes and started automatically, creating a distributed, custom-tailored version of the infrastructure. When the tests conclude virtual machines should be halted, leading to a clean testbed exhibiting the same configuration state as before running the test. This vision is detailed in the following sections.

4.3.1 Proposed Testbed Node Design

Each node participating in the testbed is configured with certain functionality yielding the necessary abstraction to use virtualised scenario evaluation. For better understanding, consider this functionality to be divided into five layers as illustrated in Figure 4.1.

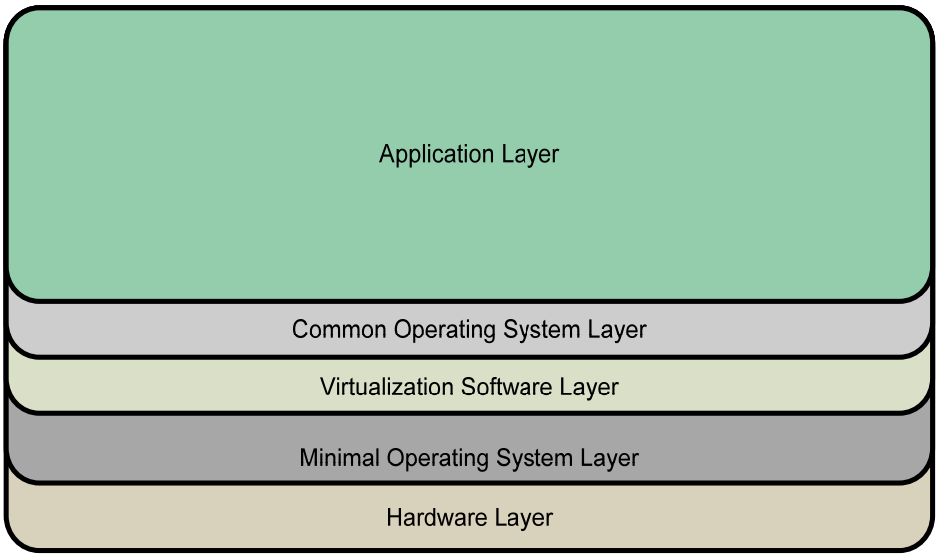

Figure 4.1: Testbed Node Abstraction Layers

The layers of abstraction are arranged in a hierarchy, with a lower layer implemented in hardware and higher layers in software. In the hardware layer all the components are physical and have real properties. Its interfaces are defined so that the various parts can be physically connected. In the software layers components are logical with fewer restrictions based on physical characteristics. The layers interact with each other through software interfaces.

Hardware Layer

The hardware layer consists of the physical devices, providing hardware resources for the testbed. Resources are processing capability, memory, mass storage and network access. These enable an infrastructure node to communicate and interact with other nodes in the local or distributed testbed. The network of the testbed is proposed to be based on Ethernet technology or Wi-Fi, potentially allowing for portability of the testbed, e.g. for demonstration purposes.

Minimal Operating System (MOS) Layer

The MOS contains just an operating system kernel (proposed to be linux based) and most needed functionality such as drivers, communication protocol implementation, and an application binary interface (ABI). The ABI provides a program with access to the hardware resources and services available in a system. The MOS is responsible for CPU, memory, device driver management, I/O (Input and Output), process, and application management. The MOS is proposed to create the interface between hardware and virtualisation software and can be considered as a very thin layer between them. It may also contain some part of the virtualisation software, which would care for execution and optimisation of some instructions operating in privileged (kernel) mode.

Virtualisation Software Layer

This layer allows several operating systems to run at the same time by portioning and sharing the underlying hardware resources. To achieve that, the virtualisation software builds and simulates an abstract and unified hardware layer upon the underlying operating system. The virtual hardware layer will be the same on each node regardless of the real hardware, i.e. the hardware layer contains the same CPU instructions, memory, motherboard, storage and I/O devices. Therefore, any running operating system depending on a virtual hardware layer can be copied, cloned and replicated without a reconfiguration and in a convenient way. Moreover, virtualisation software is able to simulate more than one hardware instruction where more than one operating system could be deployed on the same virtualisation software as depicted in Figure 4.2.

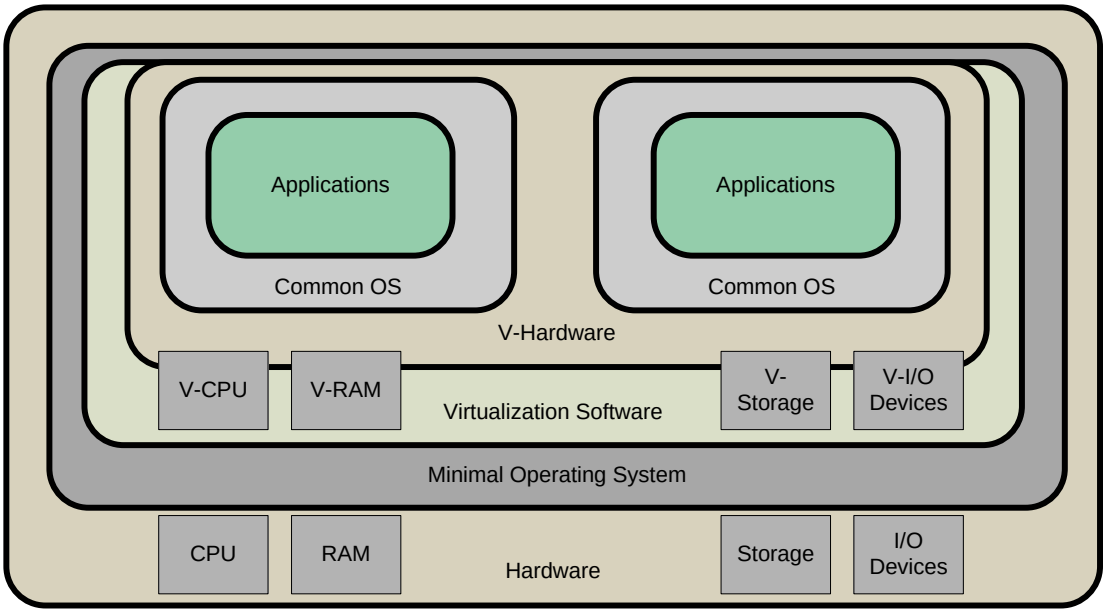

Figure 4.2: Using Two Operating Systems on One Node

Common Operating System (COS) Layer

The COS is proposed to be the place where technology developed within the CASCDAS project is executing. Due to the abstraction from the hardware it is possible to deploy different operating system such as Windows, Linux, Solaris, etc. which runs within the virtualisation software behaving like a natively executing operating system (of course there is a small penalty for virtualisation overhead, this should not be more than 20%). The OS will not see the real hardware; it just senses the virtual hardware that is simulated by the virtualisation software. The virtual hardware and the common OS build the virtual nodes. Each virtual node will be able to communicate with other virtual nodes as part of the

Application Layer

This layer contains the application scenarios that will be developed during the CASCADAS project.

Owing to the flexibility through employment of virtualisation developers would be able to experiment with several ideas by using different operating systems and deploying many concepts with minimal configuration efforts in order to compare them.

4.3.2 Virtualisation

As the utilisation of virtualisation technology is a key part of the proposed testbed infrastructure, this section gives an overview of the involved concepts, technologies and products.

Virtualisation is a broad term that refers to the abstraction of resources in many different aspects of computing. "Virtualisation is a framework or methodology of dividing the resources of a computer into multiple execution environments, by applying one or more concepts or technologies such as hardware and software partitioning, time-sharing, partial or complete machine simulation, emulation, quality of service, and many others" [Si06].

Hardware and Software abstractions permit to manage complexity in computer systems mainly by providing interfaces. For example of abstraction in computer systems, consider

the construction of a physical hard disk that is divided into sectors and tracks and abstracted by an operating system so that it appears to an application as a set of files. An opposite example would be the ability to make a set of physical hard disks appears as one logical hard disk partition through a virtualisation layer. This implies a fine differentiation between abstraction and virtualisation, i.e. virtualisation is about creating illusions whereas abstraction is about hiding complexity. The level of detail in a virtual system is often the same as that in the underlying real system without necessarily hiding details. Moreover „virtualisation provides a way of relaxing the foregoing constraints and increasing the flexibility" [SmNa05], i.e. when a system or subsystem, e.g., a processor, memory, or I/O device, is virtualised, its interface and all resources visible through the interface are mapped onto the interface and resources of a real system actually implementing it. Consequently, the real system is transformed so that it appears to be a different virtual system or even a set of multiple virtual systems.

The idea of virtualisation is not new; the concept was in existence since the 1960s when it was first developed by IBM to provide concurrent, interactive access to a mainframe computer. Virtualisation was used as an instance of a physical machine that gave users an illusion of accessing the physical machine directly [AdBa66]. Popek and Goldberg made their exploration of virtualisation in 1974 [PoGo74]. They developed a formal model of a third generation computer system in order to find sufficient conditions to determine whether a particular third generation machine can support virtualisation. According to their research formal virtualisation involves the construction of a homomorphism that maps a virtual guest system to a real host. This formal construct has then been used to characterise abstraction as well as virtualisation.

The concept of virtualisation can be applied not only to subsystems such as hard disks, but to entire machines, which are then called virtual machines. A Virtual Machine (VM) is implemented by adding a layer of software to a real machine to support the desired virtual machine's architecture.

4.3.3 Virtual Machine Concepts

Virtual machines depend on the computer architecture or the Instruction Set Architecture (ISA), which makes the division between hardware and software [LiFl02]. ISA describes aspects of computer architecture including native data types, instructions, registers, addressing modes, memory architecture, interrupt and exception handling, and external I/O. Any software build to a given ISA can run on any hardware which supports that ISA.

There are two parts of an ISA that are central to the definition of virtual machines: User ISA, containing all instructions for application processes and System-ISA, including instructions for supervisor software such as operating systems. Another important interface is the Application Binary Interface (ABI), which provides the interface between process and system space and gives access to hardware resources and services. ABI contains a set of user instructions and a system call interface, which are provided by operating systems. It allows applications to interact indirectly with the shared hardware resources after checking access privileges. A program binary compiled to a specific ABI can run only on systems with the same ISA and operating system. Virtualisation has not only to take care of the proper mapping of virtual resources or states (see [SmNa05]), e.g., registers, memory, or files, to real resources of the host machine, but also of using real machine instructions and system calls to carry out the actions specified by virtual machine instructions and system calls, e.g., emulation of the virtual machine ABI or ISA.

Process Virtual Machines

This type of VM is often referred to as the "runtime software". It is used support a guest process and to run on top of conventional operating systems. The VM support will run as long as the guest process runs. As illustrated in Figure 4.3, virtualisation software translates a set of operating system and user-level instructions composed for one platform to another, thereby forming a process virtual machine capable of executing programs developed for a different ISA.

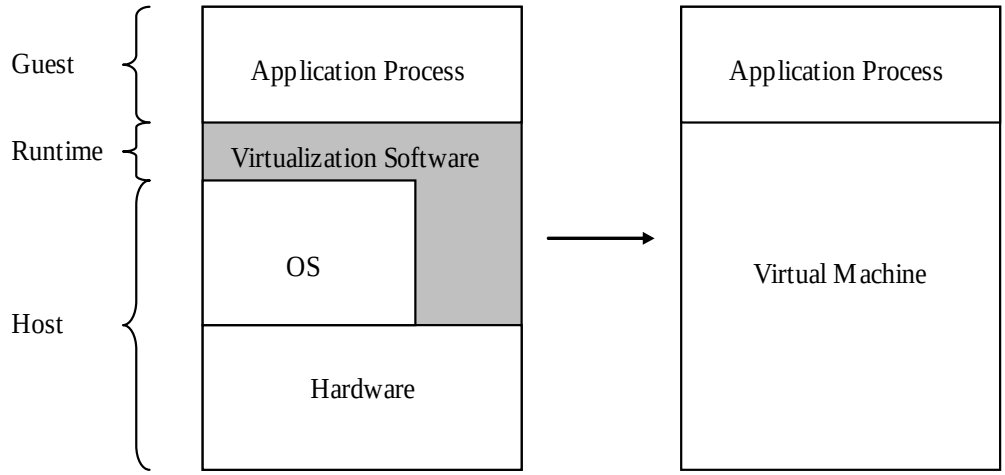

Figure 4.3: Process VM

Therefore a process VM provides user application with a virtual ABI environment for a different implementation.

Multiprogramming

A modern operating system supports multi user process, where each user process is given the illusion of having a complete machine to itself. This is called process multitasking and hardware time-sharing. In effect the operating system provides a replicated process-level virtual machine for each of the concurrently executing applications. A new virtual concept called User Mode Linux (UML) [Di06] extends this technology in order to provide more then one kernel at the same time.

Process Emulation

Process emulation is the ability of a process-level VM to support program binaries compiled for a different instruction set than the one executed by the host's hardware. For example, a software application which is compiled for x86 architecture can run on PowerPC architecture through emulation. Emulation in general is realised through two methods, interpretation, and binary translation. The interpretation method may be a slow process, requiring an interpretation of each source instruction for the native target platform. The binary translation method converts blocks of source instructions, which are mapped to equivalent functions as used in the target platform. Because this method causes a high overhead due to the translation process, translated blocks can be cached and repeatedly executed. The latter method is known as dynamic binary translation [YuJie06, KiSm03].

HLL VM

High-level Language Virtual Machines are used to achieve cross platform portability through independence from concrete ISAs. HLL VMs are based on intermediary instruction representation, e.g. bytecode that encodes instructions as a sequence of bytes. To eliminate register requirements, bytecode instruction sets are stack based and have an

abstract data specification and memory model. The Sun Java VM and Microsoft common language infrastructure (CLI) are examples of HLL VMs.

System VM

System VMs provide a complete environment enabling the installation of arbitrary operating systems and applications. Therefore System VMs need to create a set of virtualised hardware resources, such as a processor, storage resources, and peripheral devices. From the user perspective a System VM behaves like real hardware. System VMs are able to support different operating systems running on the same, single hardware platform at the same time. Dividing a single set of hardware resources among multiple guest operating system environments is a central problem of System VMs [PoGo74].

Co-Designed VM

As the name suggests, these VMs use a combination between hardware and software virtualisation, where VM software appears as a part of the hardware implementation, as illustrated in Figure 4.4. Co-designed VMs are not intended to virtualise hardware resources or to support multiple VM environments, but to provide better performance, power efficiency and design simplicity. Co-designed VMs provide binary translation in order to convert a guest instruction into native ISA. The CPU of Transmeta Crusoe is a good example of a co-designed VM. In this processor, the underlying hardware uses a native Very Long Instruction Word (VLIW) instruction set, and the guest ISA is the Intel x86. Another example is the IBM System/38 (later System/370 [Cr81]), where the aim was to minimise the gap between ISA and HLL. Further information can be found in [SmNa05, KeSe03].

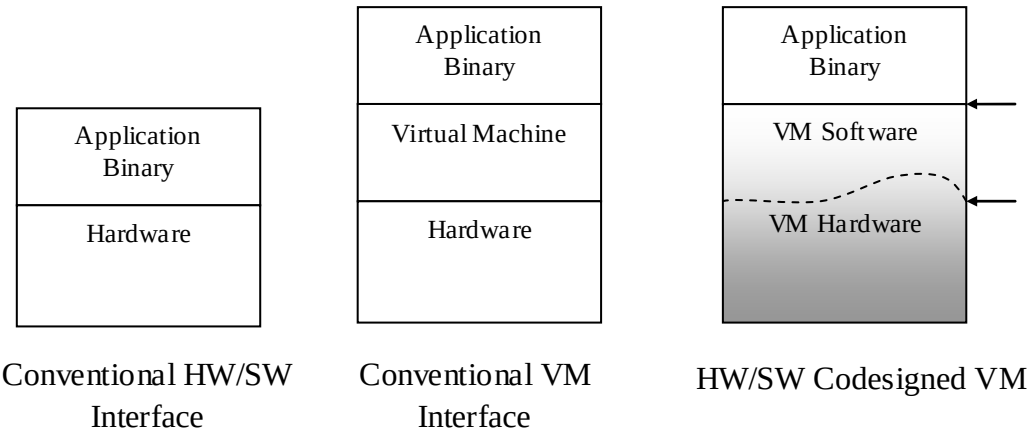

Figure 4.4:Co-Designed VM

In Figure 4.5 an overview of a possible classification of VM according to ISA and virtualisation level is given along with examples. It is proposed to use solely hardware that is based on the x86 architecture for the CASCADAS testbed, as this commodity hardware and supposed to be readily available at most partners' sites.

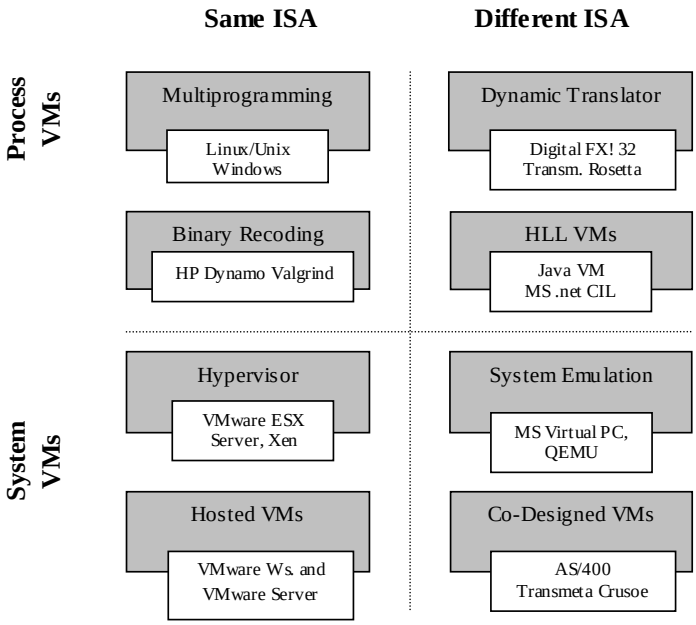

Figure 4.5: VM Classification

4.3.4 Virtualisation Products Overview

In order decide on concrete virtualisation products that may be used within the testbed the following section gives are a brief introduction to major competing products.

Xen

Xen is a free system virtual machine monitor which began as a research project at the University of Cambridge and is now commercially supported by the company XenSource Inc. The first public version was released in 2003. Xen is mainly based on the x86 architectures and supports Intel Itanium and PowerPC processors. Xen is referred to as a native virtual machine, but it can also be considered as a dual host system virtual machine concept. After some modification of the kernel, Xen runs under Linux as a host system. In order to be able to do this it uses a technology called paravirtualisation that is able to perform most of the instructions in a native way. Paravirtualisation presents a virtual machine interface to a system that is similar but not identical to the underlying native hardware, as is shown in Figure 4.6 which has been taken from [BrDr03]. Many Linux distributions, e.g. SUSE Linux already include the Xen-packages in order to make the installation easier. Kernel modifications are made on the guest operating system by Xen, whereby the guest operating system is modified to use a special hypercall ABI instead of using the features of the normal architecture.

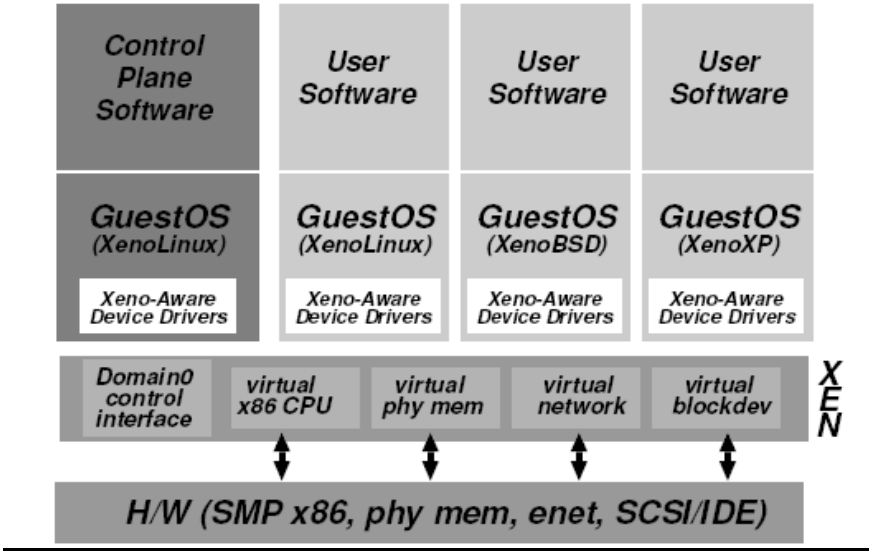

Figure 4.6: Xen organisation

Research about Xen reports in [BrDr03] that the number of lines of the Linux code base that are needed to be changed in order to virtualise the complete OS are less than 3000, or 1.36% of the total. Xen is very close to the native OS and achieves a good runtime performance (better than 90% of the original system speed across most benchmarks).

To deploy Windows XP as a guest operating system a modification of the OS should be performed. Such modification was reported possible with but is not publicly available as it involved changing Windows source code. The latest version of Xen (Xen 3.0), which at the moment of writing is available as beta is based on a hardware assisted virtualisation, where new instructions are offered to support direct calls by a paravirtualised guest OS into the hypervisor or the Virtual Machine Manager (VMM), alleviating the need to modify the guest operating systems.

QEMU and KQEMU

QEMU is emulation software written by Fabric Bellard [Be05]. QEMU implements a processor emulator and allows simulating a complete computer system within another one. It is open source and achieves a good emulation speed by using dynamic binary translation. QEMO offers two operation modes:

- Full system emulation: in this mode, QEMU emulates a full system, including a processor and various peripherals. It can be used to launch different operating systems without changing the system codes.
- User mode emulation: this mode works on Linux hosts only, in which QEMU can launch Linux processes compiled for one CPU on another CPU.

QEMU is able to emulate various hardware instruction architectures, such as x86, SPARC, PowerPC and MIPS processors. It is designed to execute on x86, x86_64 and PowerPC architectures and able to simulate a number of hardware devices, such as motherboards, different CPUs, and network devices. There are also many additional plug-ins and tools for QEMU that have been developed to provide means for simple installation and administration. Of special interest here is the KQEMU acceleration module, which is

available only on the x86 architecture and runs under Linux and Windows as host operating systems. The KQEMU Accelerator Module does nost modify the guest operating system. It is free to be used, but it is a closed source proprietary and currently free-of-charge product.

One of the major features of QEMU and KQEMU is the flexibility. They are able to execute on a wide combination of host OS and guest OS without modification of the sources of one of them.

VMware

There are many products which are associated with VMware Inc. The products considered are: VMware Server, VMware Player and in particular VMware Workstation because VMware Server and Player are based on the VMware Workstation engine [WoHa05] which supports mainly x86 architecture. For these products VMware Inc. produces various additional administration and management tools. The products in general support SMP (Symmetric Multiprocessing) systems and are able to bootstrap from FDD (Floppy Disk Drive), HDD (Hard Disk Drive), CD-ROM, DVD-ROM or Preboot Execution Environment (PXE), which allows computers to bootstrap using a network interface card [Intel03]. VMware provides Application Program Interfaces (APIs) and a Service Development Kit (SDK) for scripting and programming in Perl, C++ and Visual Basic for Developers in order to extend their products.

VMware Workstation provides a virtual machine and software tools. It allows configuring a multiple processor x86 hardware architecture. Each virtual machine instance can execute its own operating system. VMware Workstation supports virtual bridging and host only virtual network adapters. It is also possible to configure the virtual network adapter driver to use network address translation (NAT) through the host machine rather than bridging which would require an IP address for each guest machine on the host network. Besides the network adapters, CD-ROM, hard disk, and USB devices, VMware Workstation can simulate other hardware such as graphic cards and sound cards. The specialty of the VMware Workstation is the ability to take more than one snapshot of a running virtual machine. This allows users to make a backup or to do a rollback to a certain situation in a convenient way.

VMware Server is based on the VMware Workstation engine uses a client-server model to allow remote access to the virtual machines. It is also able to run virtual machines images by other VMware products and also by Microsoft Virtual PC. The VMware Server is also able to create a single snapshot copy of each separate virtual machine within the VMware Server environment. The VMware Server is a free product for up to 50 machines but it is closed source.

VMware Player is free software that can run guest virtual machines that have been created by other VMware products using the same engine as VMware Workstation. VMware Player is supposed to be not able to create a new virtual machine. It is intended to run virtual machines that have already been created on other machines.

Microsoft Virtual PC

Microsoft Virtual PC was originally created to emulate a PC in a Macintosh environment. It uses virtualisation on x86 architectures and emulation software on PowerPC architectures, but does not support the Intel dual core architecture under Mac OS X. Virtual PC was written by Connectix and subsequently acquired by Microsoft, who released the Windows version as a free product: Virtual PC is able to deploy various guest operating systems that are able to run on the x86 architecture such as Windows or Linux. It is very similar to the

VMware Workstation and specified for the use under Windows operating systems as a host. It supports bridged network mode and NAT mode and it is able to bootstrap from FDD, HDD, CD-ROM, DVD-ROM or PXE. There is also a commercial server version of Virtual PC, which is based on the server-client model and similar to the VMware Server product. Microsoft plans to integrate the Virtual PC in future Windows operating systems.

Parallels Workstation

The Parallels Workstation was released in 1999 as a commercial product. It supports PowerPC and x86 architectures and runs under Windows, Linux and Mac OS X on x86 hardware as host operating systems. Similar to the VMware products Parallels Workstation uses a hardware emulation and virtualisation approach. The new products of Parallels Workstation support the hardware virtualisation of Intel's Vanderpool Technology [Intel02] and AMD Pacifica architectures. The hardware virtualisation and implemented instructions for CPUs that are able to optimise virtualisation are examined more closely in [WoHa05]. Parallels Workstation supports bridged network and NAT mode. It is able to bootstrap from several media such as VMware products and Virtual PC, but it cannot bootstrap from PXE. Parallels produce additional products, such as Parallels Compressor Workstation and Server, which are able to import and to manage VMware images.

Products Summary

The flowing table summarises the features of the introduced virtualisation products.

Name	Xen	QEMU / Accelerator	VMware Server	Virtual PC	Parallels Workstation
Producer	University of Cambridge, Intel and AMD	Fabrice Bellard	VMware	Microsoft	Parallels Inc
Host architecture	Intel x86, Intel x86_64, (PPC planned)	Intel x86, Intel x86_64	Intel x86, Intel x86_64	Intel x86	Intel x86, Intel VT-x
Guest architecture	Intel x86, x86_64, (PPC and IA64 ports planned)	Intel x86, x86_64, ARM, Sparc 32/64, PowerPC, MIPS	Intel x86, x86_64	Intel x86	Intel x86
Host OS	NetBSD, Linux	Windows, Linux, Mac OS X, FreeBSD	Linux, Windows, Mac OS X-Intel	Windows	Windows, Linux, Mac OS X on Intel
Guest OS	Linux, BSD, Windows XP, Windows 2003	Windows, Linux, FreeBSD, BeOS,	DOS, Windows, Linux, FreeBSD, Netware, Solaris	DOS, Windows, OS/2	DOS Windows, Linux, FreeBSD, OS/2, Solaris
SMP	Yes	Yes with	Yes	Yes	Yes

Support		Plug-ins			
Mechanism	Paravirtualisation and Porting or Hardware Virtualisation (Vanderpool & Pacifica-CPU)	Emulation and Virtualisation (KQEMU) using dynamic recompilation	Virtualisation using dynamic binary translation	Virtualisation using dynamic binary translation and emulation on PowerPC	Virtualisation using dynamic binary translation
Support	No	No	no	Yes	Yes
Guest OS Speed relative to Host OS	Native	Near to native with acceleration	Near to native with VMware tools	Near to native on x86	Near to native
Additional Tools or Drivers	Integrated	Many plug-ins	VMware tools and management Tools	Management Tools	Management Tools
Usability	Command line configuration	Command line configuration	GUI configuration	GUI configuration	GUI configuration
Allocation of RAM	Yes	Yes	Yes	Yes	Yes
Guaranteed allocation of RAM	Yes	Yes	Yes	No	No
Fixed allocation of CPU power	Yes	No	No	No	No
License	GPL	GPL / LGPL	Proprietary – Free to use	Proprietary - Free to use	Proprietary
Price	0€	0€	0€	0€	50€

5 Test-beds interconnection

Three principal test-beds will be available to evaluate and demonstrate the autonomic toolkit and selected application scenarios to be developed during the next phase of the project. Initial experiments are anticipated to be conducted cooperatively, but managed independently at each location. However, the three test-beds are projected to be interconnected at a later stage of the project to create a distributed test-bed for larger scale

experimentation. This section explores possible mechanisms to achieve the interconnection.

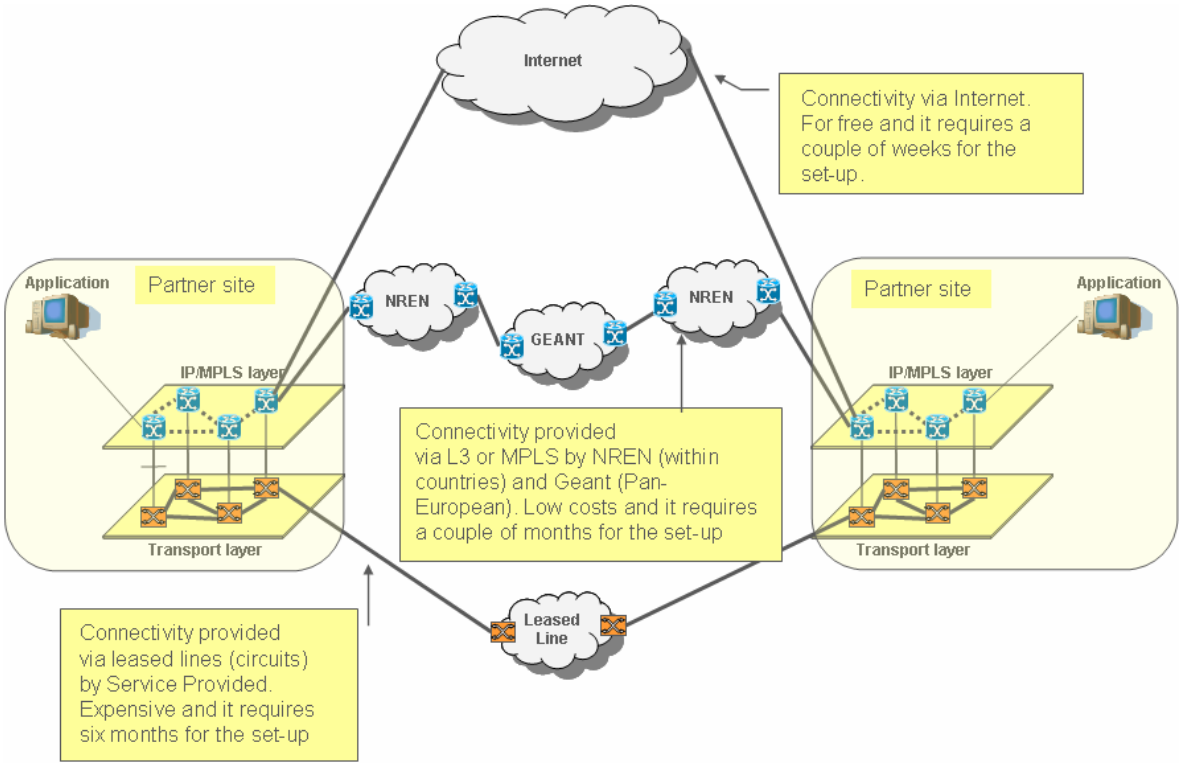

Figure 5 - Possible solutions for test-bed interconnection

5.1 Leased lines

A technical solution is to establish permanent and dedicated connections among the three principal test-beds. The solution requires local telecommunications carriers at each interconnection point to create circuits of a selected rate. The circuits will be leased to the consortium for a period of time. Having dedicated circuits to interconnect the test-beds offers several advantages: 1) consistent bandwidth links, 2) privacy with a high-level of security (e.g. against denial of service attacks), and 3) availability of telecommunications operator support and commitment for ensuring a service level. The downside of this solution is that it might require a long setup time.

Telecommunication carriers offer various alternatives for creating high-speed digital lease-lines. The most common ones are based on time division multiplexing (TDM, e.g. T1), Frame Relay and ATM circuits. In terms of equipment, routers with specific interfaces will be needed to support the interconnection via leased lines. Each site will require a router with integrated or non-integrated CSU/DSU (TDM), a serial interface (Frame Relay) or an ATM interface.

5.2 GÉANT2 and NRENs

NRENs connect, within a given country, research and educational institutions. GÉANT2 is a pan-European backbone network connecting national research and education networks (NRENs) that was created to provide a high-speed (multi-gigabit) network infrastructure to support joint research of networking technologies and services.

Test-bed interconnection via GÉANT2 and NRENs can be accomplished via IPv4 or IPv6 and QoS might be achieved with MPLS. The advantages of this solution are: 1) easily accessible, 2) large bandwidth, 3) low cost. It might require a few months to setup.

5.3 Internet

Finally, it is also possible to establish test-bed interconnection via Internet connections. The solution is of low cost and can be implemented in a very short time, as no third-party setup would be required. On the downside, the solution does not offer consistent communication channels for the interconnection and in addition, the solution might be exposed to security attacks. Nevertheless, the solution is attractive and the inconsistency of channels might create a more realistic network infrastructure to test the ability of the autonomic toolkit to adapt to changing network conditions.

One approach for interconnection via Internet is tunneling, also known as port forwarding. It consists in encapsulating private network packets in Internet packets (IPv4). This way, private information regarding test-bed experiments and remote access can be passed through the Internet. Tunneling is a simple concept that can be easily adopted and developed. However, there are as well a few well-known tunneling protocols, such as the point-to-point tunneling protocol (PPTP, RFC 2637) and the generic routing encapsulation (GRE, 2784) that can be used. These protocols also enable a certain level of security and allows for the creation of virtual private networks (VPN).

5.4 Using Virtual Private Networks (VPN)

The communication within the overall testbed would be based on the Transmission Control Protocol and Internet Protocol (TCP/IP) and use mainly the Internet Protocol version 4 (IPv4). The Internet Protocol version 6 (IPv6) may be deployed optionally if partners see a need due to test requirements. Address assignment is planned to be done through dynamic host configuration protocol (DHCP) service or auto-configuration in the case of IPv6. Communication between local testbeds is proposed to be done by employing virtual private network (VPN) technology allowing partners to deploy one homogenous address space independent of domain boundaries and geographical location. VPN technology also helps to protect and secure the distributed infrastructure.

Employing this technology also yields some negative aspects: For example as VPN traffic is encrypted, the network throughput is less than when using leased lines. As encrypted connections are end-to-end, VPN connections are more prone to Internet connection problems. Setting and administrating a VPN network may also be challenging from an administrative perspective, depending on whatever concrete technology is chosen to create the network.

5.4.1 Technology

Several competing software and hardware solutions are available to create VPNs that would suit the purpose of local testbed interconnection; following is a summary of the most widely used solutions for establishing secured connections over public internet.

IPSec	IP security (IPSec) is the most widely known protocol suite to secure IP streams and was originally designed to be implemented in hardware. It is now often used with software (for example by the OpenSwan and StrongSwan projects, which are both derivates of

	FreeS/WAN that ended in 2004 as a development project). IPSec is integrated with all recent Windows versions, Mac OS X, BSD derivates and some Linux distributions. It is also an integral part of the IPv6 protocol suite and therefore the logical choice for securing IPV6. Pure IPSec is regarded as to be complex to administrate and not working well in NAT environments ("too complex to be secure" [FeSc99]).
PPTP / MPPE	A common combination is an utilisation of the Point-to-Point tunnelling protocol (PPTP) in combination with Microsoft Point-to-Point Encryption (MPPE). This technology is integrated in all modern versions of windows and also supported for Linux, BSD derivates and Mac OS X. Encryption strength is considered as quite weak (40bit or 128bit key length).
L2TP	A different approach for securing interconnections between Windows based systems has been jointly developed by Cisco and Microsoft and is known as the Layer 2 Tunnelling Protocol (L2TP). It is also compatible with PPTP and employs IPSec to achieve a stronger encryption than MPPE. L2TP development is focused on the Windows platform; implementations for other operating systems are not in a production state.
OpenVPN	OpenVPN is an open source solution for building VPNs using Secure Sockets Layer (SSL) technology. SSL is widely used for securing connections between UNIX-like systems and offers strong encryption, as well as a flexible configuration, e.g. supporting password and certificate authentication. Implementations are available for all major platforms, including Windows, Mac OS X, all BSD derivates, Linux and other UNIX distributions. OpenVPN has support for IPV6, dynamic IP and NAT, but configuration may be complex.
Hardware	Several companies sell Hardware VPN Routers with support for IPSec and L2TP. These enable secure connections without the need for installation of software on another device and are therefore less error-prone and difficult to set up. In general this solution may be more stable than the software variant; also manufacturer support is available to deal with problems. Buying such a box may cost between 500 € – 4500 €, depending on manufacturing company and capacity / features of the hardware in question.

Considering the CASCADAS project requirements, two solutions seem to be favourable:

Solution 1: Using OpenVPN

While OpenVPN is a rather new technology, SSL has proven to be stable and is the de-facto software standard for encrypting system interconnections. From experience the configuration and administration is easier than pure IPSec, but not as easy as using a hardware solution. The source code is open, potentially allowing for extension and also fixes in the case of problems; there is also an active community supporting the product.

IPv6 is supported and the product itself is freely available under the OpenVPN license (based on GPL).

When using OpenVPN, contributing partners need to install the software on a commodity hardware machine (Hereafter called the "gateway") in their local domain, enabling secured access for the other partners using public internet.

Solution 2: Using a Hardware VPN Gateway

As the gateway machine would not to be set up proprietary, this solution will most likely be more stable and need less configuration effort that the software solution. Connections will be directly handled by the networking element, resulting in faster and more reliable connections. Support if available either separately or as part of the product package; in any case such a solution makes it necessary to spend money on specialised hardware.

Apart from the gateway machine there are requirements on the testbed network:

- Each partner's testbed network should be moved to the extranet, a so-called De-Militarised Zone (DMZ), which would greatly simplify the overall network configuration. An alternative for situations where this is not feasible would be to use VPN pass-through technology.
- Usage of static IP to give a consistent and plain topology configuration. This is not a strict requirement but as there is no need to have the testbed use dynamic addressing it's the easiest way. Once a distributed testbed exists, several dynamic overlay networks might be used on top of the static base configuration to evaluate aspects of autonomic network configuration.
- Use IP addresses that are contained in a single subnet. This will also make configuration much easier and would allow for instant multicast and broadcast capability. If v6 of P will be used, standardising on a common address prefix would be a similar step.
- Provisioning of sufficient bandwidth. As concrete requirements on the testbed in terms of throughput, delay, etc. are not fixed at this stage of the project there are no clear numbers on this. Nonetheless we should assume that every partner will be able to utilise an infrastructure that is able to satisfy the test scenario needs.

5.4.2 Topologies

Regarding the testbed topology there are two proposals, either using a fully meshed topology or using a centralised "star" layout.

Meshed Topology

In a fully meshed topology every partner will be directly connected to everyone else, please see Figure 5.1 for an example of a fully meshed network.

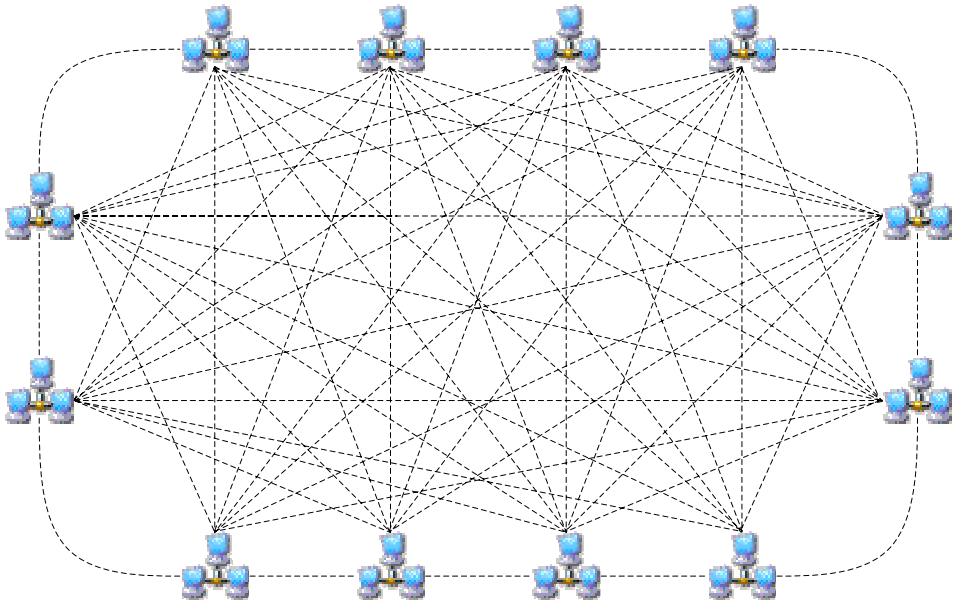

Figure 5.1: A fully meshed network of 12 nodes

The advantages are as obvious as the disadvantages: a direct line ensures the best possible connection speeds between all partners, but the more partners join the network the more complex and time consuming administration and connection establishment will become. This topology is best employed using dedicated gateway machines (e.g. VPN routers) at each partner as the load and routing effort is equally high at each partner.

"Star" topology

Using this topology, one partner has to take a leading role and the other partners are connecting to this central server (hereafter called "network head"). See Figure 5.2 for an example with a hypothetical "partner 2" as the network head.

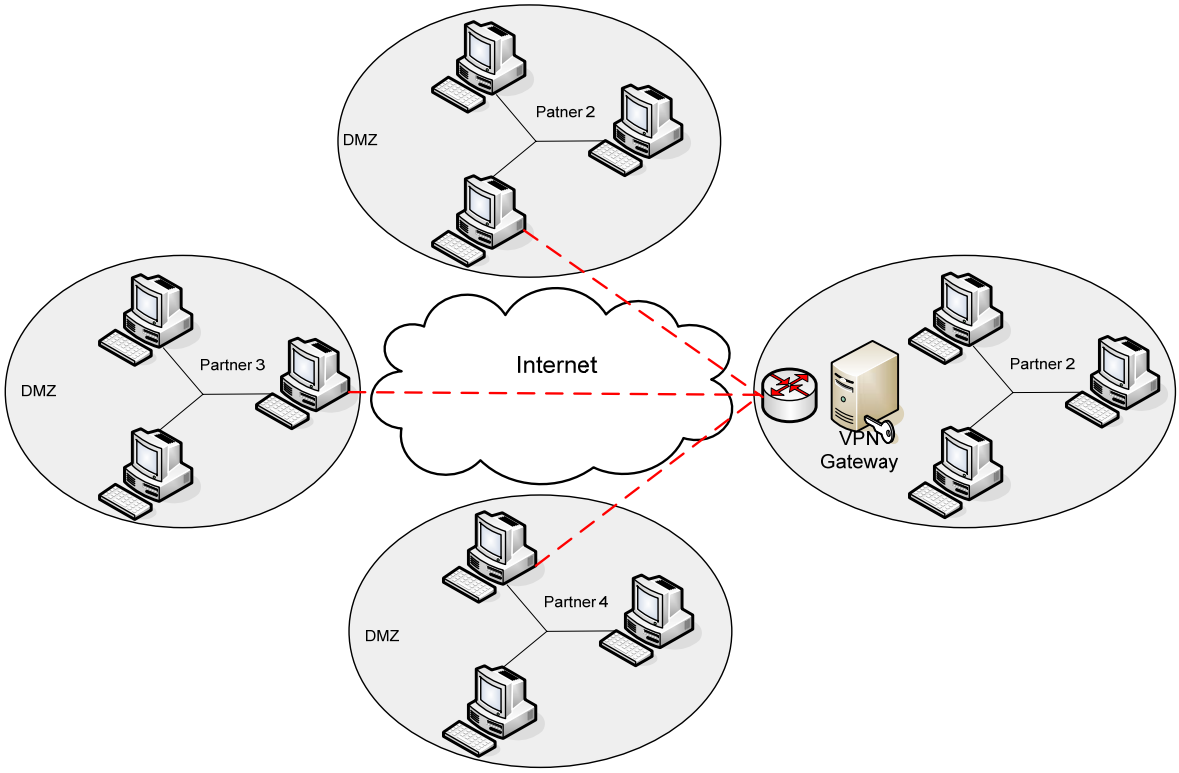

Figure 5.2: "Star" topology for a VPN network

This topology is easier to understand, deploy and administrate and may therefore be more stable. As only the network head needs to manage all connections and route packages load is high at this single gateway, whereas other partners only need to invest modest effort to keep the network running. This topology supposes a stable and appropriate gateway machine at the network head as this obviously is a single-point-of-failure for the testbed network. Connections may expected to be more delayed than in the fully meshed scenario, as all traffic needs to be routed through the central gateway.

6 Roadmap

The development of the autonomic toolkit, application scenarios and test-bed activities aimed at demonstrating the Open Autonomic Service Framework is a horizontal task that requires integrated work across all work packages of CASCADAS. The following is an estimated of what can be delivered on the dates outlined. It is worth noticing that because of the research nature of the project, dates might deviate and alternative or extra features may be incorporated to the roadmap in the future.

6.1 Summary of Planned Activities

The anticipated activities related to the development of application scenarios and demonstrators are summarised in Figure 4. The activities are planned from month 13 to month 30 of the CASCADAS project.

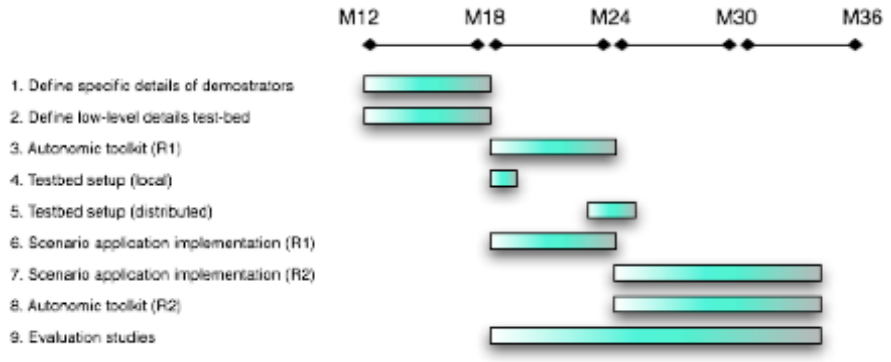

Figure 4 –Planned activities related to testbed and application scenarios

NOTE: An elaboration on the activities follows (to be revised according to WP1 and WP6 activities and draft planning).

1. Definition of low-level details of demonstrators. Consists in extracting and analysing the low-level requirements to be developed from each application scenario starting from the work reported in D6.1 part A. The purpose of this activity is to ensure that such requirements are consistent with the services and features of ACEs and to identify potential demonstrations to be developed (e.g this might include additional software for the visualisation of specific parameters).
2. An extended study of the test-bed setup will be conducted during the indicated period.
3. First release of autonomic toolkit. The first release of the autonomic toolkit is expected to happen not after M24 and it will be coordinated by T6.3 with the participation of all technical WP.
4. Independent test-bed setup. Test-beds at each location will be independently setup to have them ready to test imminent software releases of the autonomic toolkit and application scenarios.
5. Test-bed interconnection. We expect that the initial development and testing activities will require only a local test-bed for their initial experiments. However, by means of test-bed interconnection we expect to make available a larger test-bed to CASCADAS researchers for larger scale experimentation and demonstration.
6. At least one demonstrator based on an application scenario will be developed during the indicated period.
7. Additional features will be integrated in to the initial release of the demonstrator and potential extra demonstrators will be released. During this phase, it is expected that specific software to support demonstrations will be developed and tested.
8. Evaluation studies. Evaluation studies will be conducted by means of analysis, simulation and test-bed experimentation concurrently with other phases of the

project with the purpose of providing adequate feedback to the design of the autonomic toolkit and applications.

9. Second release of autonomic toolkit. Following the initial release of the autonomic toolkit, work will be concentrated on improvements suggested by WP while evaluations occur.

7 Conclusions

CASCADAS aims at developing an autonomic toolkit based on distributed self-similar components that will serve to construct future services, which will exhibit self-properties to simplify their creation, usage and management. A key milestone of CASCADAS is the design and deployment of a distributed test-bed to evaluate, improve and demonstrate a prototype of the autonomic toolkit.

This document has described the foundations for the materialisation of a CASCADAS test-bed that will be used both to evaluate key ideas and to demonstrate the use of the autonomic toolkit for the construction of selected application scenarios. The goal of the work has been the identification of design, development and deployment needs, by addressing the principal requirements and obligatory parameters for usability and flexibility.

The document has depicted specific use cases of selected application scenarios to illustrate some of the key issues that we intend to demonstrate in a test-bed. Moreover, the document has analysed the main phases of test-bed design and requirements, as well as the main structure. Solutions to two key problems were addressed. The first problem is related to the limited number of nodes available for experimentation. The solution resorts to virtualisation to increment the number of possible nodes hosting ACEs. The second problem is related to test-bed interconnection. The requirements and various techniques to achieve test-bed interconnection have been evaluated. It is expected that the selection and implementation of one interconnection technique will take place at a later phase of the project.

Finally, the document has provided a roadmap for future activities, which are necessarily integrated to the work of other WP, the implementation of the autonomic toolkit and the development of selected application scenarios. Such activities will take place during the next months of the project.

Appendix "What is available"

The distributed CASCADAS' test-bed will physically consists at least of three local test-beds (potentially interconnected via an Internet VPN) located at the premises of FOKUS, UNITN and ICL. The following is a description of the available equipment at each location.

FOKUS

The local Fokus testbed consists of a single management server and 9 small devices, everal of them mobile. Most of the testbed is also portable.

The equipment in detail:

- 5 x Portable Mini-ITX PC's VIA EPIA-MII12000 with a x86 compatible processor with 1.2GHz frequency, 1GB RAM, 40GB HD, 100Mbit LAN, PCMCIA Wi-Fi
- 2 x Ultra-Mobile PC's OQO Model 1+ with 1GHz Transmeta Crusoe processor, 512MB RAM, 30GB HD, Wi-Fi, Bluetooth
- 1 x MacBook Pro with 1.86 Intel Core Duo processor, 2GB RAM, 120GB HD, Wi-Fi, Bluetooth
- 1 x Nokia 770 Internet Tablet with 250MHz TI OMAP 1710 processor, 64MB RAM, Wi-FI, Bluetooth
- 1 x Dell Rack Server with dual 64Bit Intel Xeon 2.8GHz, 2GB RAM, 2 x 250 GB HD, 2 x Network cards

UNITN

The UNITN test-bed consists of access points deployed in the Faculty and serve both students and professors. The authentication system has been designed and implemented by our team, and supports user localisation and other profile information. It allows for multiple authentication sources, and its interface should now be general enough to support the implementation of some carefully designed self-* property, for instance to let clients assist devices with low computing capabilities in choosing the best authentication procedure, or to enable a wireless client to help the authenticator itself by asserting other nodes' identities under its own responsibility. The widescreen displays are part of a student information system, and are placed in the halls of our buildings. They are driven by MacMini boxes. While our group doesn't own the displays, we can use them to some extent for experimenting context-aware information systems. In particular, we are now integrating them with a bluetooth interface to detect the proximity of users' cellphones and react by displaying the most appropriate information. Once an early platform is deployed, evolution towards a testbed for the Behavioral Personal Advertisement scenario shall be considered.

The equipment in detail:

- 6 Intel-based Servers
- 3 Desktop PCs (Dell)
- 20 Access Points (Cisco, 3Com, D-Link, LinkSys)
- 2 Routers (LinkSys)
- 3 LAN Switches (Allied Telesyn, D-Link)
- 5 Palmtops (iPAQ)

ICL

The ICL test-bed consists of a number of mobile and stationary computers. The computers run a version of the Linux operating system modified by our team to investigate self-aware networks.

- 82 Rack PC Intel P4 2.4 GHz (single core, hyper-threading)
- 512 MB RAM, 5 - 12 Ethernet interfaces Linux (Debian)
- 3 Desktop PC Intel P4 2.4 GHz (single core, hyper-threading)

- 512 MB RAM, 4 Ethernet interfaces, WiFi 802.11g Linux (Debian)
- 1 Wireless access point
- 7 PDA H5500 Linux Familiar 0.8.2
- 4 Fast Ethernet LAN switch (24 ports)
- 3 Cabinet, patch panel

UNIMORE

The UNIMORE test-bed consists of hardware and software that will be specifically devoted to experiment with situation-aware algorithms.

- 2 sensor network kits "CROSSBOW MICAZ"
- 2 short range RFID readers
- 1 long range "Alien" RFID reader
- 4 Compaq IPAQ PDAs
- 2 Bluetooth GPS receivers
- various WiFi access points