BIONETS

WP 3.1 – REQUIREMENTS ANALYSIS AND ARCHITECTURE

VTT, TUB, INRIA, UNIHH

Service architecture: requirement specification and concept definition

Deliverable D3.1.1

Reference:	BIONETS/VTT/WP3.1/1.1
Category:	Deliverable
Editor:	Janne Lahti (VTT)
Author(s):	Juhani Latvakoski, Jyrki Huusko (VTT), David Linner, Ilja Radusch, Heiko Peffer, Stephan Steglich (TUB), Francoise Baude, Ludovic Henrio (INRIA), Daniel Schreckling (UNIHH), Francesco De Pellegrini (CN)
Verification:	Davide Mandato (CN)
Date:	18/06/2007
Status:	Final
Availability:	Public

Executive Summary

The goal for this deliverable is to provide the initial service architecture definition and to identify the fundamental service requirements for the BIONETS application scenarios, in which the services are no longer isolated and hardly reachable, but these autonomic services form social, self-organizing networks. The deliverable provides also initial guidelines for research work in BIONETS subprojects SP2 and SP3, giving a high-level overview of the necessary system components and their interactions. Each component is described to the level of detail that is needed to understand its function in the specification.

The service architecture needs to cope with the issues such as the service management, content and terminal adaptation, service discovery and security aspects as well as the evolution of services and life-time. In addition, the service should be able to work properly on top the network architecture. This kind of "disappearing network" environment, where availability of resources, network conditions and user requirements can change dynamically and where the co-operation of nodes is not guaranteed and disconnected operations are common, generate challenges for the service architecture. Based on the requirements rising from the different application scenarios, evolution, service autonomy and underlying network architecture, a new BIONETS service framework, which is able to cope with the challenges in efficient and cost-effective way, will be defined.

The deliverable can be divided into two parts. In the first part application scenarios are illustrated based on the application scenario analysis in the work package WP1.1. After that, several service specific requirements are announced for the applications scenarios, and some fundamental requirements for BIONETS service computing environment are given. Respectively the second part of the document illustrates the BIONETS service architecture principles and provides an initial service architecture definition for BIONETS SerWork (service and network) architecture. During the project time the service architecture definition and also the requirement specifications rising from the application scenarios, will be reprocessed and expanded iteratively.

Document History

Version History

Version	***Status***	***Date***	***Author(s)***
0.1	*First Draft*	*05.06.06*	*Janne Lahti, Juhani Latvakoski, Jyrki Huusko*
0.3	*Draft*	*07.06.06*	*Janne Lahti, Francesco De Pellegrini*
0.4	*Draft*	*15.06.06*	*Janne Lahti, David Linner*
0.5	*Draft*	*14.07.06*	*Janne Lahti, David Linner, Francoise Baude, Daniel Schreckling, Ludovic Henrio, Ilja Radusch*
1.0	*Final*	*09.08.06*	*Janne Lahti, David Linner, Jyrki Huusko*
1.1	*Amendment*	*18.06.07*	*Jyrki Huusko, Janne Lahti, David Linner, Francesco De Pellegrini, Markku Tahkokorpi*

Summary of Changes

Version	***Section(s)***	***Synopsis of Change***
0.1	*Not Applicable*	*None - first draft, table of contents*
0.3	*2*	*Scenarios added*
0.4	*3,4*	*New table of contents, sections complemented*
0.5	*1,2,3,4*	*Sections complemented*
1.0	*all*	*Harmonization of sections, Executive Summary and Future Work sections complemented*
1.1	*2*	*Emphasis of the evolution aspects in the proposed application scenarios*

Note

Reviews after final document delivery (Version 1.0) to the project may or may not result in modifications to the document. If modifications post review is necessary, then the first version of the resultant document is 1.1.

Contents

1. Introduction 5

2. BIONETS Application Scenarios 7

2.1 Scenario 1: BIONETS coupled Guidance system 7
2.1.1 Evolutionary and bio-oriented aspects in the scenario 8

2.2 Scenario 2: Wellness Use-Case 8
2.2.1 Evolutionary and bio-oriented aspects in the scenario 10

2.3 Scenario 3: Virtual Guides in a Pervasive Computing Environment 10
2.3.1 Evolutionary and bio-oriented aspects in the scenario 12

2.4 Scenario 4: BIONETS for handicapped people & Home Nursing 13
2.4.1 Evolutionary and bio-oriented aspects in the scenario 14

3. Requirements 16

3.1 Requirements for BIONETS Service Environment 16
3.1.1 Requirements analysis for Scenario 1 16
3.1.2 Requirements analysis for Scenario 2 17
3.1.3 Requirements analysis for Scenario 3 19
3.1.4 Requirements analysis for Scenario 4 20
3.1.5 Summary of Requirements 20

3.2 Service Architecture for the BIONETS Computing Environment 22
3.2.1 The Computing Environment 23
3.2.2 Basic Objectives of the Service Environment 23

4. BIONETS Service Architecture Principles 23

4.1 Nodes and Services 24
4.1.1 Technical Service Definition 25
4.1.2 Service Types 25

4.2 Service Framework 26
4.2.1 Runtime 26
4.2.2 Runtime Configuration 27
4.2.3 Service Comprehension 27
4.2.4 Request Delegation 28
4.2.5 Autonomous Service Composition 29

4.3 User-Driven Service Generation 30
4.3.1 User Request 30
4.3.2 Generic Service Request 30
4.3.3 Abstract Service Implementation 30

4.4 Security 30
4.4.1 Security Requirement Description 31
4.4.2 Identification 31
4.4.3 Authentication 32
4.4.4 Access Control 32
4.4.5 Integrity 32
4.4.6 Reputation System 33

4.5 Autonomic Services 33
4.5.1 Autonomic Functions & Self-Management for BIONETS Services 33
4.5.2 Self-management 34
4.5.3 Self-optimization/self-configuration 35
4.5.4 Self-monitoring/Self-repair 36
4.5.5 Self-protection 36

4.6 Evolution 36
4.6.1 Adaptation of the Service Architecture 37
4.6.2 Evolution Strategies 37
5. Next steps and future refinements 38
References 39
Terminology 42

1. Introduction

The motivation for the BIONETS project comes from the emergence of pervasive computing environments, characterized by a huge amount of nodes which can exchange information with each other via wireless links. These environments will be characterized by three main factors that call for novel communication, networking and service provisioning paradigms:

- Scalability: the system will be composed of millions of devices, arising clear scalability problems for the underlying networking infrastructure;
- Heterogeneity: the system will be composed of devices extremely different in their architecture complexity, power requirements and computational power, ranging from RFIDs to smart-phones and laptops;
- Complexity: the system should be able to show self-organizing behavioural features, in order to radically solve the complexity issues related to the management of such large-scale network.

One of the main aims for the BIONETS is to define a new SerWork architecture, which incorporates the disappearing network architecture and autonomous services, and which is able to utilize and benefit from the multidisciplinary approach in fulfilling the networking and service provisioning paradigms. The SerWork architecture could be illustrated e.g. with the critical paths presented in Figure 1.

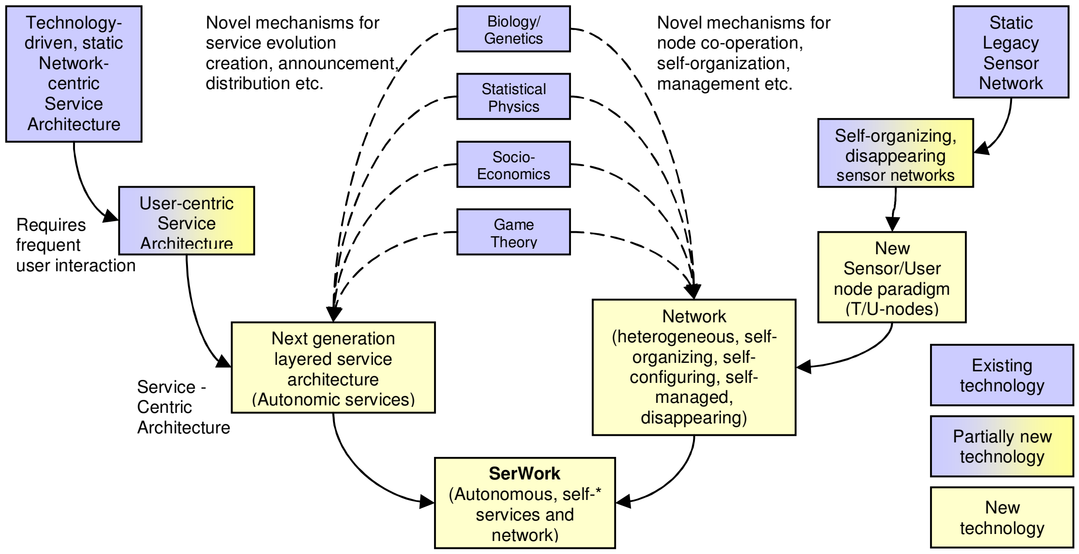
Technology-driven, static Network-centric Service Architecture
Novel mechanisms for service evolution creation, announcement, distribution etc.
Biology/ Genetics
Statistical Physics
Socio-Economics
Game Theory
Novel mechanisms for node co-operation, self-organization, management etc.
Static Legacy Sensor Network
Self-organizing, disappearing sensor networks
Requires frequent user interaction
User-centric Service Architecture
New Sensor/User node paradigm (T/U-nodes)
Next generation layered service architecture (Autonomic services)
Service - Centric Architecture
Network (heterogeneous, self-organizing, self-configuring, self-managed, disappearing)
SerWork (Autonomous, self-* services and network)
Existing technology
Partially new technology
New technology

Figure 1. SerWork Critical Paths.

Future Pervasive environments raise issues like scalability, heterogeneity and complexity, which need to be answered also in the BIONETS SerWork architecture, in terms of:

- operational functions this means disconnected operations.

- network architecture: combination of T-Nodes, U-Nodes, (APs).
- information forwarding: opportunistic forwarding for U-Node to U-Node communication, and push or pull for T-Node to U-Node communication.

From the logical standpoint, we may divide the node entities in BIONETS into two broad categories, T-Nodes and U-Nodes. The role of these nodes is expressed in terms of generation/processing/consumption of information, whereby information is intended to be the one injected into the network (intended as a system) from the environment (which, broadly speaking, represents what is external to the system).

We have:

- T-Nodes, which are the logical entities that act only as a source of information. Example of T-Nodes will be RFIDs and sensors, but this is not limiting. T-Nodes are used to gather information from the environment.
- U-Nodes are the logical entities that *consume and process* information. This is mainly intended for running *services*. Example of U-Nodes will be smart-phones and laptops, but, once again, this is not limiting. The user interfaces with U-Nodes only; however not all U-Nodes interface with a user.

The classification is based on purely logical roles, so that a device may act as a T-Node for a given period and then “upgrade” to the role of U-Node. Different logical entities can coexist in one single device, so that, e.g., a smart phone with a temperature sensor comprises a T-Node (in that there is a unit that generates data interacting with the environment) and a U-Node (in that there is a powerful computational engine that can run some services owing an adaptation capability) etc.

The basic way BIONETS deal with scalability issues is by giving up the connectivity requirement. In other way, BIONETS networks will be almost always disconnected. In BIONETS all operations should be performed on the basis of an intermittent connectivity, whereby the terms “connectivity” does not mean, in general, ability to join a backbone network, but just to communicate with other devices. As a consequence, no client-server paradigm is present: all nodes are basically “equal” (i.e., peers) and exchanges are based on localized interactions. BIONETS exploits opportunistic forwarding schemes, in which the mobility of the devices is exploited to spread the information in the network. The BIONETS network principles are described in more detail in [BIODelivD111].

In order to utilize efficiently the defined logical T/U-node architecture also a new service architecture, which is able to cope with the issues such as the service management, content and terminal adaptation, service discovery and security aspects as well as the evolution and life-cycle from creation to extinction, is needed. The goal for this document is on one hand to provide preliminary high-level specification for such architecture in BIONETS SerWork architecture, and on other hand to provide initial service requirement specification for the work in the various BIONETS SP1, SP2 and SP3 research tasks.

This deliverable is organized as follows. Section 2 presents several possible application scenarios for BIONETS project, which have been specifically analyzed from the service point of view, in order to gather several high-level requirements for the service architecture. The requirements arising from the different application scenarios for BIONETS services are described in the Section 3. Section 4 presents general principles for BIONETS architecture and describe the first initial BIONETS service architecture, including e.g. the considerations for autonomic services, evolution in service architecture, service layering approach and security. These initial guidelines will be the basis for further work in the BIONETS work package WP3.2: the architecture description and requirements will be refined during the project, based on the research results gained in WP3.2 and in the other parallel work packages.

2. BIONETS Application Scenarios

This section presents selected BIONETS application scenarios, which will be used as a starting point for the service requirement analysis. These scenarios will evolve and change and the requirements based on these scenarios will also evolve accordingly.

Whenever possible, the way scenarios are described is made with the following idea in mind: to illustrate in which respect a bio-inspired evolution of the described system may be useful, or, better, may be a good way for the entities to adapt in order to fulfil their objective. Additionally, it might also be the case that only a bio-inspired evolution would permit such adaptation.

2.1 Scenario 1: BIONETS coupled Guidance system

This scenario describes a navigation software that leverages contextualized habits and shared knowledge of a local group of users, and determines the best routes to destination. Thus this will be done not just by means of customary shortest path, but introducing more complex requirements which may involve bargain opportunities, and so on. In sight of this, the metrics used to perform the guide should be adapted to the changing habits of the user and to the peculiar requests of the individual

Currently, the Traffic Message Channel technology, a digital channel used to provide silent, coded messages to in-vehicle applications in order to display route and traffic information in a user's native language[1], allows GPS car navigators to leverage broadcasted information about road traffic/weather conditions so as to reroute itineraries in order to work around traffic jams or certain areas.

In the present scenario, local information may still represent an additional service, when the granularity of the data pertains to small districts in an urban area. But, further specialization is provided coupling navigation service and user needs.

The idea is that by coupling the user needs to the traditional requirement of *finding a route to a destination,* we will generate several novel services and algorithms as well. In practice, the novel set of combined services involving a guidance system are effectively explained by queries of the form:

- “This is my shopping list: find the most interesting deals based on local, real-time advertisements diffused in the surroundings, and guide me to efficiently visiting most convenient shops. Also, find me the optimal free parking spaces in the surroundings of the identified shops”
- “I am a person bearing handicap: I need to find the parking slot reserved for handicapped person that is located the closest to my destination”

All the above mentioned examples will leverage information retrieved from the environment. In particular, the T-nodes enter the picture in different forms: in the

[1] “Radio Data System-Traffic Message Channel (RDS-TMC) standardisation is well advanced. [As of this writing,] ENV 12313-1 'Coding protocol for RDS-TMC using Alert C' is currently being upgraded to an EN ISO standard (14819-1) through an enquiry and formal vote. ENV 12313-2 'Event and information codes for RDS-TMC' has been submitted for similar upgrading as EN ISO 14819-2. ENV ISO 14819-3 'Location referencing for Alert C' is being submitted for parallel ENV ISO ballot in a few weeks.
ENV 12313-4 'Coding protocol for RDS-TMC using Alert C with Alert Plus' has been approved in formal vote. PrENV 12313-5 'Location referencing for Alert Plus' is almost finalised.” (Source: http://cordis.europa.eu/telematics/tap_transport/deployment/standards/keytopics.htm)

shopping example, they will be represented by advertisement devices, in the handicap support will be sensors revealing parking slots for the handicapped person. Local data collected by T-nodes will be stored during the search, but their persistence should be matched to the guidance applications running on top of other U-nodes.

This scenario is meant to leverage the peculiar communication architecture of BIONETS, where local information can be leveraged in order to introduce novel functionalities in already existing applications. The user interface of the service can furthermore improve over time, based on best practices shared within the local community. Furthermore, where buses or taxies run on a regular basis in certain areas of the city, the diffusion of data crunched by the GPS navigation software could leverage this “layer” of existing U-nodes in order to spread efficiently environmental data through the U-nodes community.

2.1.1 Evolutionary and bio-oriented aspects in the scenario

The application of evolutionary and bio-oriented paradigms in this scenario will be targeted to the potential level of complexity required by the guidance application. In practice, guidance operations will be customized to user needs and the changes in the environmental conditions. To this extent, traditional optimization techniques would be not suitable to deal with a huge state space composed of unpredictable user requirements and the environmental changes. Hence, the application of Genetic Algorithms and Evolutionary Algorithms will be useful in order to achieve optimality of the solutions provided to the user.

In particular, we distinguish *short term adaptation* techniques in the case of changed input from the environment: this of course might imply the change in the metrics employed to determine the route and/or the selection of a best fitting algorithm within a set of pre-assigned solving methods. The novel metric/algorithms will be used to determine optimal solutions according to the user needs and the environmental input. Conversely, the system would adequate over time, and thus on *a longer time scale*, to the specific user requests. For example, the modification of a particular route selection algorithm and/or the use of a given metric will progressively match the typical pattern of a user movement and habits. In this case, the composition of a route selection algorithm based on evolutionary principles will adapt the metrics to the daily schedule of the user and the requirements in the driving at different hour times. This optimization can be efficiently solved via evaluation of fitness metrics and evolving the composition of the modules involved in the route determination. The semantics underlying composition of the applications, anyhow, would be transparent to the user, since they would derive from the composition of existing applications. In order to evaluate the fitness, typical metrics should be employed, such as for example the delay to reach the final destination. In all the above cases, the benefit for the use of this application will be enhanced performance of the guidance application and the possibility of customizing the offered services.

2.2 Scenario 2: Wellness Use-Case

Susan is an enthusiastic cyclist. She’s planning to attend a cycle race with a couple of her friends. The evening before the race she opens her laptop at home and starts her PersonalCoach-Service. Her PersonalCoach–Service is also able to adapt to a variety of other sport activities and is capable of evolving based on the user needs and usage history. Susan communicates to the PersonalCoach-Service that she is going to participate to certain cycle race on the following day and the Service automatically enrols her to the race using available Internet connection (AP-node). The Service automatically also downloads related information (map, weather *forecast,…) about the race and also predicts based on previous experience that Susan is going to need a CyclingCoach-Service on the following day and that she is going to use her SmartWatch-device (U-node) instead of her laptop at the race, so the Service migrates itself to the SmartWatch along with the CyclingCoach-Service.*

On the following day, Susan wears her SmartWatch and cycling clothes and the SmartWatch automatically detects the sensors in Susan's clothes and automatically configures the Body Area Network (BAN). At the racing location Susan mounts her bike and starts the race with her friends. They plan to co-operate and cycle the race as a group. When Susan is riding her bike the Body Area Network configures itself with the bikes sensors, creating a Personal Area Network (PAN). The other cyclist in the group also have their own PersonalCoach-Services in their U-nodes, so the cyclists' PersonalCoach-Service detect that there are other peer services in vicinity and, based on previous information, find those reliable, and hence decide to engage a co-operation and information exchange with those peer PersonalCoach-Services. It might even happen that one of the other users does not have any CyclingCoach-Service installed, so an instance of that service migrates from one of the cyclists' U-nodes to that user's U-node.

At the start of the race, hotspots offered wireless connection for the PersonalCoach – Service. The PersonalCoach- Service is able to get for example the latest weather forecast or road condition information for the user. It can benefit also from distributed computing for calculating e.g. energy consumption, oxygen saturation and possibly other services requiring lots of computing power. After a couple of kilometres the wireless network suddenly disappears, so the underlying network adapts itself to ad-hoc communication and the PersonalCoach- Service also adapts itself to the available communication mode and starts to use more localized data: for example it starts to do the calculations in a SmartWatch and uses other U-nodes to convey data.

After the race, when Susan is back home, the PersonalCoach-Service migrates again to Susan's laptop and analyses all the collected data from the race and presents the results to Susan, also suggesting about what to eat, how to stretch muscles and how to train to get better results in the future.

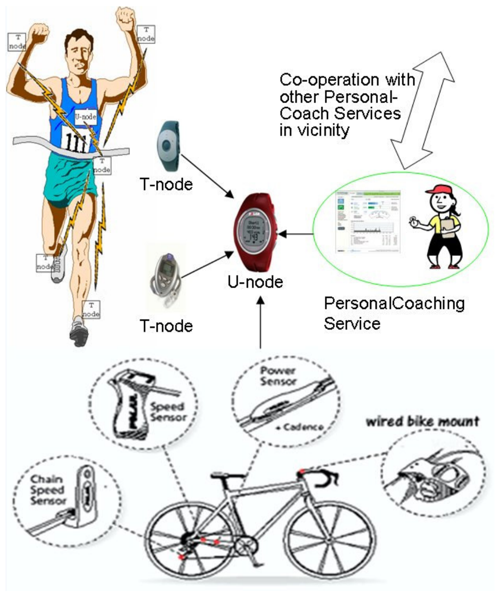

Figure 2: Wellness Service scenario

2.2.1 Evolutionary and bio-oriented aspects in the scenario

This section demonstrates in more detail how the evolutionary aspects of BIONETS services could be applicable to the Wellness scenario.

The Wellness scenario presents how the BIONETS service the *PersonalCoach* helps and guides the user in his/hers everyday training activities. The *PersonalCoach* –service is a Service Individual providing the user interface to the user and is composed of other service components each providing specific operations to the main service.

The main bio-oriented issue within the Wellness scenario is the adaptation of the *PersonalCoach* –service to the available context information and especially to the needs of the user. The adaptivity of the service in the scenario can be divided into two cases reflecting autonomous reconfiguration and progressive improvement of the BIONETS services. With short-term adaptivity, the service acts pro-actively or reacts just-in-time on changes in the environment. ([E.g. [dis]appearing devices and services, changing availability of users, etc.); aims at the realization of robustness.

The short-term adaptivity is needed in the scenario e.g. when Susan mounts her bike and the *PersonalCoach –service* configures itself with the bikes sensors, or when the other user *PersonalCoach –service* lacks the specific functionality of CyclingCoach -service and the it adapts itself in order to meet the user needs better by initiating a migration of CyclingCoach –service from another *PersonalCoach –service*. From the user point of view the short-term adaptivity or evolution of the service can be totally transparent. E.g. in the first case when adapting the service to the bike sensors or underlying network connections the evolution might lead to semantically equivalent service from the existing service, therefore the fitness evaluation cannot be based on the user feedback and must be based on a algorithm measuring and evaluating some kind of real physical values. In the second case the level of evolution is higher and it generates services with functionally that is completely new to the user. In this case the fitness of the new service can be evaluated based on the direct user feed back.

The long-term adaptivity is needed in the scenario e.g. when the PersonalCoach–Service needs to adapt itself to new sports. In the beginning when the user starts to use the service the PersonalCoach–Service contains basic functionalities which are common in most sport events, but when the user has used the service for a long time with specific sport events the service adapts itself to match better the needs of the user. For example, Susan is enthusiastic cyclist, so her PersonalCoach–Service is adapted itself over time to match better the users need by evolving to contain cycling specific functionalities and has deprecated some functionalities never used by Susan. This long-term adaptation is based on ‘Learning’ of alternative approaches to handle the user request. In order to match better the changing needs of the user the service can utilize e.g. exchangeability of services, modification of service implementations and reconfiguration of parameters. The service progressively increases the overall adequacy of the system response by the use of evolutionary principles.

The Evolutionary and bio-oriented aspects of the Wellness scenario discussed above enables to have services, which can handle both rapidly changing environmental conditions as well as slowly changing needs of the user.

2.3 Scenario 3: Virtual Guides in a Pervasive Computing Environment

The scenario addresses a choice of features that should be realized by future computing environments supporting the user's every-day life. The described system support and system autonomy may require concepts beyond the scope of BIONETS in some points, but nevertheless should help to identify requirements and challenges of a beneficial user-centric service environment. The objective is an evolving, pervasive computing system, i.e., an environment that is supposed to dynamically integrate services and features of locally embedded and globally widespread devices and applications, while forming a ubiquitous, coherent computing system around the user.

Tom is on business trip; he just left the customer's office and is now on his way to the underground garage, where he has parked his car. He is carrying several smart devices and sensors with him, for instance a small mobile terminal with a touchpad, an intelligent watch that controls his health status, a headset, and a personal gateway used to access different wide-range radio networks (UMTS, GPRS, Wi-Fi, WiMAX, ...). Together, these devices form a Personal Area Network (PAN) around him.

It is sunny summer evening, so, walking down the street, Tom spontaneously decides to go for a sightseeing tour. He formulates this as a task for his computing environment in natural language, while either using the mobile terminal or speech recognition to do so. The environment establishes an Internet connection and searches information about city's places of interest, especially for historic sites and monuments, since Tom is interested in architecture. Finally, several famous objects are presented to the user, also including information concerning their distance from the Tom's current location. Moreover, the system proposes to display a street map showing the locations of listed places of interest on a public display the user will pass by in a few seconds. The system does so since the user does not like scrolling street maps on the small display of his mobile terminal. Tom agrees and stops in front of the display, while his computing environment takes care of integrating the display and showing the offered information, which is obtained from third party services. The user chooses one of places of interest. Tom's agreement to use the alternative display suggested by the system is taken as feedback and helps the system to transparently evaluate the appropriateness of comparable solutions potentially satisfying the user's need.

Bob, an old university mate of Tom lives in this town; Tom has contacted him before setting off to this trip, and they arranged to meet. Since Tom's friend usually works till late in the evening, Tom still has enough time for sightseeing. Hence, he decides to call his friend to make out details of their meeting. The system recognizes that Bob is not available for calls at the moment, and hence delays the call until Bob becomes available again, by notifying Tom accordingly.

A few minutes later, Tom is already near the underground garage. Bob becomes available and the system proposes to set up the call now. Tom agrees, the call is established, and he can finally talk to Bob. At the same time, Tom enters the hoist of the basement garage. The computing environment realizes this fact as it knows Tom's current location from positioning sensors and location meta-information, again obtained by third party services from the Internet. Furthermore, the system knows that hoists shield wireless networks, but discovers a Wi-Fi access point in the hoist being installed there to bridge this shortcoming. The computing environment prepares a seamless handover of the call. Bob suggests meeting later in the evening at a nice restaurant located in the downtown and Tom is going to make a table reservation there. When the call is finished, Tom asks his computing environment to find the restaurant and make the reservation. The system asks him for the concrete time. Tom answers and the system successfully makes the reservation. Now, Tom is going to notify Bob on this fact; he creates a short voice message. When the *message is ready to be sent, Tom appears to be out of range of the access point in the hoist and there is no connectivity to other networks in the basement, but due to the low-priority of the message, the user's computing environment decides to delay the transmission of the message for a few minutes. As the user enters his car, the car is integrated with the user's computing environment and the destination information, i.e., the location of the place of interest is loaded into the navigation system of the car. When approaching the destination, the car's computing environment searches for nearby car parks where places are still available and leads the user on his way there (see §* ***Error! Reference source not found.****).*

In the historic downtown of the city, T-nodes are placed near every object of interest, offering information on the objects. When Tom approaches the chosen place of interest, the nearby T-node contacts his mobile terminal, so that object's history and description are presented to the user. Tom likes the sight of the building so much that he decides to shoot a photo of it and to send it to the interactive message board at home.

This scenario considers sensors placed in the clothing of the user and/or in his physical environment, as well as simple devices such as light and air conditioning as T-nodes. Mobile terminals, car navigation systems, public displays, and so on are supposed to be U-nodes. The services represent features of the single devices that form the user's computing environment. Thus, the services abstract device functionalities while the T-nodes and U-nodes provide interfaces to access those services.

2.3.1 Evolutionary and bio-oriented aspects in the scenario

This scenario aims at a service-oriented computing system that integrates most different types of devices and functionality. Basically, the scenario outlines two types of use cases, illustrating the capability of the system to adapt in short-term manner as well as in a long-term manner to changing conditions in the computing environment. While short-term adaptivity becomes for instance visible in the handover of the user's call from a wide-area infrastructure network to a small hot spot network as he enters the hoist, long-term adaptivity is for example required to approach system responses that correspond to users' preferences and environment context.

In this regard, the challenge of long-term adaptivity is the generation of knowledge about service interoperability. Even if interoperability can be enabled on a "technical level", i.e. interface are compliant or could be made compliant by adapters, interoperability would need to be enabled on semantic level to determine in which situations two services can be combined or under which conditions one service can substitute another one. A situation is defined by the context of the user and environment, for instance current location, ambience, individual preferences, mood, informational state, or situation of socially related people. However, the support of service interworking requires knowledge about services, or more exactly, knowledge about the situation-dependent composability of services, substitutability of services, and the best configuration of each service in an orchestra of services. In contrast to the classical approach of addressing this challenge, i.e., forcing service developers to create extensive semantic descriptions for services, this scenario is based on the system's capability to create, verify, evaluate, update, and reject knowledge about service interworking.

In particular, the scenario addresses use cases that build on concepts to a) create semantically *equivalent* services from existing services and to b) create semantically *appropriate* service from existing services. We see evolution as one promising biological concept for the implementation of these concepts with regard to the given scenario. For example, the call over area-wide wireless networking infrastructure and the temporal substitute of communication service provision by the access point in the hoist of the basement garage can be regarded as a semantically equivalent substitution. The public displaying service and the display of the user's mobile phone can be considered as semantically appropriate, since both provide the same functionality, but under differing non-functional properties.

In this scenario a user is consuming BIONETS services for communication, orientation and entertainment reasons. Service evolution is present in the system adaptation to environment conditions. At the moment there is no adequate technical solution in place to handle the aimed level of adaptivity. Evolutionary concepts are supposed to help filling this gap. To evaluate the appropriateness of system responses, the user is given the opportunity to rate the system behavior by graphic or audio interfaces. In return, the user benefits from the individual support for everyday tasks. Audio-visual and haptic interfaces are also used to issue service requests. In parallel, successive data gathering may help to anticipate the need for support by services. Life-time of services and data may vary in the given scenario between few hours and several years.

2.4 Scenario 4: BIONETS for handicapped people & Home Nursing

This section describes the idea of an application scenario in which BIONETS type of system could be utilized to (i) support disabled persons with their everyday activities, and also (ii) how the BIONETS could enhance the work done by people taking care of elderly and other disabled at their homes.

There are numerous examples in which modern technology supports disabled people. Usually, the users of this technology will have to adapt to their new equipment. The idea of this scenario considers a long time scale in which services running on prosthesis, implants, or other supporting equipment such as hearing/vision aids or wheelchairs learn to adapt to the user needs and the environment. This environment may be handicapped accessible buildings, buses, or houses and apartments the handicapped person uses or in which the user lives. Imagine a blind person who has to move to another city, who wants to go on holidays in another country, who changes her job, or who simply moves within a city. This person will be faced with a completely or at least partially different environment. Different real-life services this person needs to use will be available at other locations. Schedules of buses will be different. The buses themselves will be different, their seating arrangements, their route, the way they announce next stops - if they are announced at all. The route this person has to take to get to a specific location will be different. Thus, a person will have to get used to the specific environment. An assisting service that learns the level of help a person needs and the adaptation this individual experienced may be very helpful. Why is this? Once adapted to the new realities, a supporting service may become annoying. So, in certain locations and in a specific context the service may adapt and provide no or only appropriate information. This is a very subjective configuration and may differ a lot for different people. On the other hand, a default service may also need to adapt in another way. Some people may find it hard to adapt to its new environment. Additional information and help will be needed. According to the user feedback such an assisting service will have to adapt. This feedback may be provided passively in terms of sensor readings or actively by direct interaction with the service. As an example you may imagine a bus which is equipped with sensors which may track the position of a blind person. Depending on the tracking results the service may decide whether it has to improve its directions or not.

To make the services associated with people running more efficiently, services (running on U-Nodes of course) that are situated in buses, buildings, or apartments will need to adapt to the services used by handicapped people and provide appropriate information. Even the exchange of whole services or fragments thereof, which were evaluated to be more helpful for people with the same or similar disabilities, may be exchanged. T-Nodes will provide readings from the surrounding environment, which have been made accessible for handicapped people. Additionally, sensors may also be associated with a person, so the T-Nodes may simply move with the handicapped individual. We do not know how far this idea can be developed but we think that there is some potential if this example is expanded to various types of disabilities and use cases.

An example of how to support professionals assisting elderly / disabled people is *home nursing*: *this is a typical work carried out in different locations (people's homes). The nurses typically have an office where they meet and schedule daily tasks. The main activities carried out at the patients' premises are: taking care of patients' basic needs like preparing food, possibly ordering more foodstuff, giving bath and, last but not least, taking care of giving the correct medicine to the patient.*

Due to the dynamic nature of the tasks (nurses change daily, change of patients, their medication changes etc) it would be very beneficial to have a common distributed calendar/task list among the nurses, which would also include the reporting of the work done and the general status of the patients.

To improve the reliability of medication, U-nodes can check if the medication given is indeed the one that was supposed to be given to the customer. This would be achievable by having an indication of patient's identity and the type of medication by using T-nodes.

Due to the confidential nature of medical information, this should be securely stored and transported when needed.

In case of unexpected changes, the nurses should be able to delegate tasks to each other on peer-to-peer fashion. Task description would contain all the necessary instructions for performing the task even in the case when the substituting nurse is meeting the patient for the first time. Task descriptions will evolve over time base on the feedback nurses give to the underlying service.

2.4.1 Evolutionary and bio-oriented aspects in the scenario

This section attempts to analyse the possible areas within BIONETS scenarios for handicapped people & Home Nursing where evolutionary services could be applicable.

2.4.1.1 Bionets for Handicapped People

The main bio-inspired issue within the first set of use cases of this scenario is the adaptation of the assistance service to the situation of the user. This adaptation depends a lot on the specific disabilities of the user and should develop based on the feedback from the user.

The level of assistance depends also on the situation of the user i.e. if the situation is new or if user is engaged in a repetitive action like daily commute.

In a new situation the service itself need to go into learning phase so that it can learn the purpose of the user in as unobrusive way as possible. This phase most propably requires a human guide as a teacher for both the user and the service.

2.4.1.2 Bionets for Home Nursing

The second use case is more challenging from evolutionary point of view. There are many issues within the use case which need to be “fixed” like the fact that people preferably need to have food every day and medication should be given regularly and according to the prescription.

One aspect of the use case that could adapt rapidly is the real-time re-scheduling of the appointments with the customers. As mentioned in above description one central issue of the concept is the distributed calendar system among nurses. If the system is able to optimise the routing and scheduling of the work based on the real work done – even in situations where unexpected changes happen - the quality of the service can be improved.

Similar adaptation as in the first use case could be used for giving instruction to the nurses especially for various medical/caretaking procedures. The service within the U-node could detect if the procedure required is new to the nurse or if there is a long time since she/he has done that procedure previous time. In such situation the service could remind nurse on key issues related to the procedure. It woulsd also be possible to evolve the descriptions of various procedured according to individual situation of the cutomer and her/his preferences via feedback from the nurses. This would also enhance the quality of the end service to the customer and give support to the nurses in areas they might feel themselves insecure.

3. Requirements

One of the key aspects of the BIONETS is the emergence of pervasive computing environments, in which a huge amount of nodes are interacting through wireless links. But no conventional client-server paradigm is present: information exchanges are based on localized interactions, in that only local connectivity is, in principle, present. Therefore BIONETS services should be based on location based information, requiring fast response times, user interaction, and dynamic, on-line information filtering.

First we need to determine in which aspects BIONETS services will differ from the global mobile services, what features can make BIONETS a widespread architecture:

- In BIONETS services we do not need to store and process the information centrally, it is enough to do that in the localized mobility area.
- Since the users need up-to-date local information, we need to collect and process the information on the spot, what will properly describe the actual status of that micro-environment.
- The changes of the environment can happen very rapidly (for example Wi-Fi hotspots) and affect a huge number of users in that mobility area, so we need to track very carefully the actual conditions of the environment.
- The information should be important for many users, making them interested in the further transmission of the information and spreading the service.
- Service adaptation: self-improvement based on user feedback, or on the popularity of the services.

3.1 Requirements for BIONETS Service Environment

3.1.1 Requirements analysis for Scenario 1

Shortly put, this scenario presents a novel breed of autonomic context-aware services. As this scenario suggests,

- *Services must be composable within the framework of the guidance application*: in particular, their respective evolution strategies should be either merged, or should be able to collaborate so as to drive the adaptation of the composed service (e.g. the result of this composition could be that some shops, which are of relevance according to the given shopping list are not presented to the user as they are situated in too remote areas and would introduce too large delays to reach the destination).
- *A service may be client of other services in order to achieve its own goal*. For instance, in the scenario 1, the “shopping service” needs to use the services of both a “route planer” and a “handicap parking slot finder”.

From a security point of view, we can identify the following rough requirements the described service needs to fulfil:

- The described *services will have to maintain the privacy of the user as sensitive data is involved in the task to be accomplished*.
- Autonomous combination of services may create vulnerable and misbehaving services not complying with any policies. *Protection mechanisms against vulnerabilities as well as policy violations of new services will be required*.
- Security mechanisms in this scenario will also *need to deal with new and changing security requirements imposed by newly created or modified services*.

- In the case of reservation and/or automatic payment for a user, *the system will need to be able to authenticate to a service, or services will need to be able to authenticate on behalf of an appropriate user*.
- Services distributing information about available resources may be malicious and thus provide erroneous information, causing damage in terms of energy consumption, waste of time, higher expenses, etc. To judge whether the distributed information is eligible the *services will need some feedback about the authenticity of this information or about the level of trust other users put in the source of this information.*

A route planer, guiding a vehicle of a non-handicapped person to a parking place which is explicitly reserved for handicapped people will cause financial loss. Thus, on the one hand, *this scenario requires some mechanism assessing the performance of your service ("good" behaviour). On the other hand, there is a need to be able to identify the source of trouble*.

- In the scenario described the service may perform correctly but the environmental information may be wrong. *Some type of non-repudiation will need to be involved in critical applications*.
- Spam, which does not only cause annoying messages but which may also consume scarce resources, is also a possible threat in this scenario. Appropriate countermeasure for BIONETS will be required in this scenario.

3.1.2 Requirements analysis for Scenario 2

The scenario 2 presents a personal coaching service in a BIONETS system assisting users in their sport activities. The main idea of this scenario is the adaptation of the service to user's social networks and personal context, and also to changing environmental conditions, like existing network connections or lack of connections, thus enabling the user to access the service with any device and location.

- Before the service can assist the user, the user needs to locate/find first the service somehow. In the BIONETS environment, where there is vast number of services available, the manual service discovery and installation procedures might not be feasible. Therefore the *BIONETS services need to be easily/automatically discovered by the user.*
- It could be even possible that the *service finds the user by using user's contextual information and (semi-)automatically deploys itself to the user's device.* It could be also possible that the service autonomously emerges based on the data the user has, the features of the environment, and past history of interaction with other networking entities.
- Based on these requirements it can be seen that the BIONETS service should be *autonomous with respect to service creation, announcement, discovery, and self-deployment*. It is obvious that these requirements also induce appropriate security demands ensuring that these sensitive operations are carried out in a secure way. Service creation will generate new security requirements for newly created services, change security policies, and may also create services which handle user information inadequately (see also Section 3.2.1).
- The service should also be able to *extend its functionalities to other devices or deploy itself on a number of redundant nodes*. This ensures a better availability of service building blocks, but at the same time increases resource consumption. This self-deployment could be *service migration* to other U-nodes, whereby the whole service logic is moved to another U-node, or it

could be *a distributed approach where only fragments of the service logic are transferred to other device* or it could be a *service replication where a service makes a copy of itself and sends it to other node(s)*. Secure distribution and execution of distributed code and/or code fragments is a logical requirement.

- Because the environment where the sport activity takes place and the available sensors varies a lot, depending on which sport the user is practicing, the *service needs to be highly adaptable.* The service has to handle a massive amount of sensor information available in the environment and to find the needed reliable information source.
- Also the type of the available sensor information varies a lot, it can be, for example, body area sensors (heart rate, blood pressure,…), personal area sensors (speed, location,…), or some third party sensor (weather, temperature,…). Data may also have a broad variety of security requirements. Especially body area and personal area sensors provide a lot of information that are subject to a wide range of privacy concerns. If the sport activity takes places, for example, in a running lane there might not always be a network connection available, so although the service should use network connections when available, *the service should perform reliably even in situations whereby no end-to-end network connections are available*.
- The *service needs also to adapt itself to the changing needs of the user*. For example, if the user needs certain function that cannot be delivered by the service
- The *service needs to engage a co-operation with other services to fulfil the needs of the users*. The activity might happen in some sporting event, whereby also other users are nearby doing similar activity. In these cases the service should be able to utilize the user's social networks and to engage co-operation with nearby similar services. The services could, for example, share information or functionalities with each other. At the same time it has to be ensured that this co-operation does not penalise its user. As an example you may think of a competition in which functionalities are shared between opposing teams.

In order to be able to survive and evolve in these kinds of resource constrained, dynamic and heterogeneous environments without relying on a centralized control, the *BIONETS service needs to be highly autonomous*.

- In BIONETS the *service can be constructed out of other services and resources*, in a way that one service offers interface for the user and uses other services to do the actual functionalities. Therefore the service will need detailed knowledge of its components, current status, resources available in nearby nodes, and all connections to other systems to govern itself autonomously.
- The autonomic service needs also to be aware of other services in nearby devices. It needs to find better ways to interact with neighbouring services and *it should be able to know identities of those services and if they are trustworthy and reliable* enough to engage co-operation with.
- It's not enough that the service functions in a normal situation and perfect conditions. *It needs to be able to discover problems or potential problems, and then find an alternate way of using resources or reconfiguring system to keep functioning smoothly*.

When the service is first installed onto a device, it might function nearly optimally, but as the needs of the user and the context of the environment change with time, the *service needs to change itself in order to keep working nearly in optimum state*.

- *The service should be able to configure and reconfigure itself in varying and even unpredictable conditions*. Of course, reconfigurations must be performed in such a way that it does not cause damage to other services or the user itself. This damage may be indirect by for example leaking personal sensitive data or direct by deleting important data to keep other services running. Thus, it is required to survey or control execution of services and checks the compliance of these reconfigurations with existing policies.
- Because the needs of the user will change over time, the service also *needs to develop itself not only to match better the requirements from the user*, but also to function in a more efficient and secure way. In the scenario 2 the service initially only handled one sporting type, but later it could evolve to handle other sporting types. This development shows a learning behaviour, where *the service is able to evolve based on the user needs and usage history*. This development could be quite fast adaptation, for example, getting new functionalities by downloading code fragments from other services or it could be slower adaptation, by changing the parameters of the program. This slower adaptation can be achieved in form of evolution, where the user's needs and behaviour constitute the primary feedback to the evolving services: in this way the user is able to trigger service reshaping and improvement.

3.1.3 Requirements analysis for Scenario 3

Scenario 3 addresses the human's everyday-life in computer-aided environments. Therefore, the scenario comprises different tasks that form an exemplary picture of desirable environment behaviour.

- The central aspect of the scenario is the *integration of very heterogeneous services into a coherent service environment*. For instance, mobile user terminals provide messaging or visualization features to the user, while fixed devices like sensors, network access points or public displays provide general information, data transfer services or Graphical User Interfaces. Global networks enable the access to classic web services and allow for example the reservation of tables in a restaurant or provide users with tourist information.
- To gain value-added functionality in those integrated service environments, *services usually need to be composed in reasonable manner*, as suggested by the sequential execution of multiple services in the scenario.

Additionally, composite services are required to *evolve in an ad hoc fashion* so as to address users' novel tasks on demand. The functionality of those composed services is limited by the functionalities of the single services within the service environment. When regarding service execution and service composition, the context of all involved objects is of major interest. On the one hand, the context of the user represents implicit knowledge that may be critical for the comprehension of the user's goals und consequently important for service executions. For instance the user's current location is beneficial when running a virtual tourist guide, while user preferences are useful to select where to present a street map. On the other hand context information helps to determine the importance of services dependent on the situation. A wireless access point to the Internet, for example, would only be beneficial for a user if the access point was in the range of his mobile terminal.

The third important aspect that may be obtained from the scenario is the need for adaptive interaction with the outer environment, i.e., usually the user.

These interactions are supposed to achieve two objectives.

- First, to gain information that is required to accomplish a certain goal and is not explicitly given in the knowledge about the context, i.e., information that is identified after the original service request. In the scenario the system asks the user for the time when to reserve the table, as the user may have forgotten to provide the system with the time in the original service request.
- Secondly, the service environment, as well as the services executed within it, need to be monitorable. Otherwise there is no way to recognize whether there are any actions in progress or not. In the scenario sending a message is delayed until the required resource, i.e., a network access point becomes available again.

However, regarding services as basic elements for the accomplishment of tasks on behalf of the user, evolution may be utilized as approach to find a composition of multiple services that promises to best accomplish a requested task.

3.1.4 Requirements analysis for Scenario 4

From this scenario it appears that it may be useful for a *service to provide a history of the adaptation that it has run since its creation*. It would be nice to have such a history expressed in a comprehensible way for any service. *A service instance may be replaced by another instance, of the same type or not, and in this case, at least offering a similar functionality*. Or an already *existing instance may be totally reconfigured in order to consider a drastic change* (e.g. in scenario 4, the service instance dedicated to help the handicapped person to take the bus will need to be reconfigured as this person has changed her job). The use of the history could be relevant in order to quickly configure the new instance, still taking into account the user preferences, needs (handicaps as in scenario 4), which had been reflected by the corresponding previous adaptations. To sum up, when adaptations of a service pertained to user preferences and profiles, it might be the case that the same preferences still apply on a new instance. Also, scenario 4 showed the relevance of having sharing of services which were evaluated to be helpful within a given category of users. Again, having a history could help new instances to quickly configure so as to reach this quality of service expressed as a good fitness.

- Privacy is one classic requirement in networks used by different entities. As we see from scenario 4 sensible (medical) information must be kept unshared across a subset of services (all patients), but be shared across some others (shared between a given patient and the subset of nurses that take care of her). This requires a capability to identify which data sent or received by services should be secured.

3.1.5 Summary of Requirements

This section summarizes the requirements from the scenarios presented within the previous sections. The Table 1 lists these requirements with ID number and name and gives a short description of each requirement along with a pointer to the scenario(s) where each requirement is found (column “Scen”), whether it is a mandatory or optional requirement for BIONETS service (column “Mand/Opt”) and the dependencies from other requirements (column “Dep”).

ID	Name	Description	Scen	Mand/ Opt	Dep

1	Automatic Service discovery	BIONETS services need to be easily/automatically discovered by the user	2	opt	
2	Automatic Service creation	The service doesn't always need to be created by a user. Services can be also created automatically by the system.	2	opt	
3	Self-deployment	The service should be able to extend its functionalities to other devices or deploy itself on a number of redundant nodes	2	opt	4,5
4	Service distribution	BIONETS services should be able to run distributed on many nodes.	2	opt	5
5	Adaptability to environment	The service has to be highly adaptable to the changes in the environment	1,2,3, 4	mand	6
6	Scalability	The service has to handle a massive amount of sensor information available in the environment	2,3	mand	
7	Connectivity	The service should be able to function in a situation where no end-to-end network connections are available	2,3	mand	
8	Adaptability to user needs	The service needs also to adapt itself to the changing needs of the user	1,2,3	mand	
9	Co-operation	The service should be able to engage co-operation with other services in order to fulfil the requirements from the user	1,2,4	mand	5,22
10	Social-networks	The service should be able to utilize the social-networks of the user, in order to offer better service quality to user	2	opt	5,9
11	Autonomicity	In order to be able to survive and evolve in these kinds of a resource constrained, dynamic and heterogeneous environments, without relying on a centralized control, the *BIONETS service needs to be highly autonomic*	2	mand	
12	Service composition	Service can be constructed of other services and resources in a way that one service offers interface for the user and uses other services to do the actual functionalities	1,2,3	mand	9
13	Identity	Services should have unique identity	2,4	mand	
14	Self-recovery/repair	The service needs to be capable to recover from any potentially problematic situations. It needs to	2	opt	

		be able to discover problems or potential problems			
15	Self-configuration/optimisation	The service needs to change itself in order to keep the near optimum state	2	opt	5, 8, 11
16	Evolution	The service needs to develop and evolve itself not only to match better the requirements from the user but also to function in a more efficient and secure way	1,2,3	opt	5
17	Identification	Required for authentication	1,2,4	mand	13
18	Authentication	Services and users are required to be able to proof their identification	1,2,4	mand	17,13
19	Access Control	Required to restrict access to data, functionalities, service	1,4	mand	17, 18
20	Integrity	Required to ensure that a service known in a network is available in an unmodified version	3	mand	19
21	Reputation System	Monitoring the behaviour of a service and inferring some knowledge about its possible future behaviour.	3,4	opt	5,13
22	Context-awareness	The selection of appropriate services as well the service execution needs to be aware of the environment context (e.g. the user context)	2,3	mand	5
23	On-demand service composition	Value-added services may be composed (created) on demand as answer to a complex tasks	1,2	opt	2,3,8, 11
24	Autonomous information gathering	In case of a leak of information required to answer a service request the service environment should try to gather these information autonomously from the user or 3rd sources	3	opt	5,22
25	Integration	Service from all domains need to be integrated, i.e., services form mobile and fixed hardware devices as well as service from large-scale system as the web	3	opt	
26	Monitorability	Activity of services and service environment need to be monitorable all the time	2,3	opt	

Table 1: Requirements for scenarios

3.2 Service Architecture for the BIONETS Computing Environment

3.2.1 The Computing Environment

The BIONETS Computing Environment is intended to be basically composed of two types of computing nodes, which are referred to as T-Node and U-Node. A T-Node is an abstraction of physical devices with hardly limited physical resources, i.e. few storage capacity, slow processors, limited networking capabilities and poor or even no own energy sources (e.g. sensor motes or RFID tags). In contrast, U-Nodes are abstractions for computing devices with substantial physical resources, for instance mobile user terminals, desktop computers or even home entertainment devices. Additionally U-Nodes may provide user interfaces to the BIONETS computing environment. On behalf of the user, nodes of both types form ad hoc ensembles to perform user task in a cooperative manner. Since some of the underlying devices are supposed to be mobile, or at least connected through wireless networks, the ensembles of nodes are supposed to be highly dynamic. Consequently the BIONETS service architecture should be designed fault tolerant, especially with regard to disappearing nodes and intermittent communication channels.

3.2.2 Basic Objectives of the Service Environment

In general, the service environment allows the user to interact with the computing environment. These interactions comprise:

- The retrieval of data from the environment,
- The manipulation of data stored within environment
- As well as the initiation and the control of process executions.

Data retrieval may for instance address information about the physical surrounding of the user, any processing results created by the environment, data left by others users, or application dependent data.

Data manipulations are intended to change the settings of the environment, i.e. the configuration of devices abstracted by the nodes.

Initiation and control of process executions may be utilized to handle data streams or actions performed by the environment.

However, a service is supposed to combine these three types of interactions in a reasonable manner in order to perform a given user task. Moreover, the service environment supports active and reactive service provisioning. Active service provisioning is initiated by the user while requesting a certain task to be performed immediately. In contrast, a reactive service provisioning is effected by a conditional user request. If the user request contains one or more trigger conditions, service provisioning is suspended until the conditions are fulfilled. Those conditions may for instance be bound to the information about the user context, like current location or ambience.

4. BIONETS Service Architecture Principles

This is a preliminary description of the BIONETS architecture based on the first draft of the BIONETS Service requirement analysis. This architecture description is to be intended as work in progress and subject to be refined in next iterations

4.1 Nodes and Services

The BIONETS computing environment is intended to be basically composed of two types of computing nodes, which are referred to as T-Nodes and U-Nodes. A proper definition of this separation can be obtained from Deliverable 1.2.1 [BIODelivD121]. On behalf of the user, T-Nodes and U-Nodes form ad hoc ensembles to perform user tasks in a cooperative manner. Since some of the underlying devices are supposed to be mobile, or at least connected through wireless networks, the ensembles of nodes are supposed to be highly dynamic. Consequently, the service architecture has to address two different types of computing nodes with very different physical capabilities. The nodes are not integrated in a homogeneous service architecture based on the lowest common dominator. The architecture rather integrates both types of environment resources while separating the tasks they need to cope with according to the task's complexity. This separation visualized in Figure 3 and explained below.

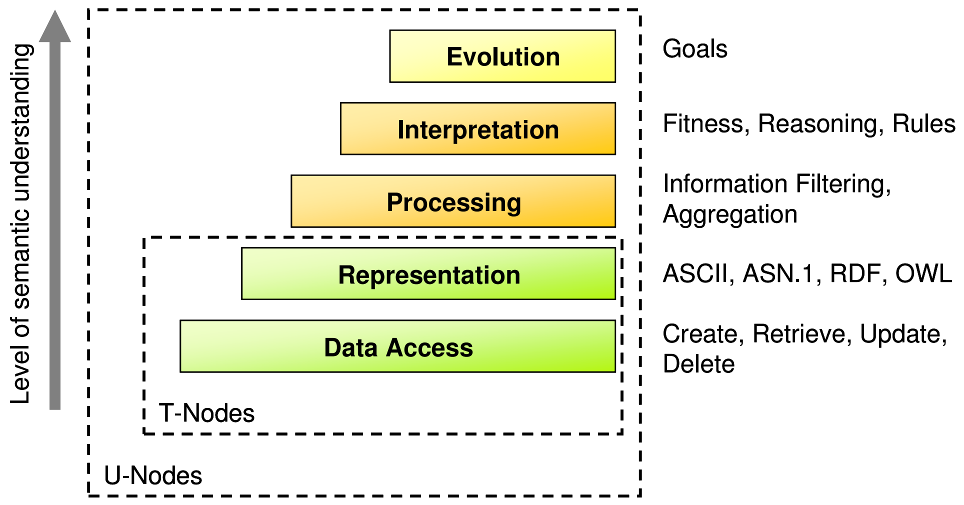
Level of semantic understanding
Evolution
Interpretation
Processing
Representation
Data Access
T-Nodes
U-Nodes
Goals
Fitness, Reasoning, Rules
Information Filtering, Aggregation
ASCII, ASN.1, RDF, OWL
Create, Retrieve, Update, Delete

Figure 3: Layered view on services

1. *Data Access*. (T-Nodes and U-Nodes) Basic operations in this layer are:
   - create: new information on a node.
   - retrieve: gathering data from a node,
   - update: modify existing information,
   - delete: cancelling existing information,

   These operations are applied to *buffers*, meaning that the information arriving to the data access layer may be originating from a sensor reading or from a network card. Both cases are considered in the data access layer.
2. *Representation* of data. (T-Nodes and U-Nodes)
3. *Processing.* (U-Nodes) Information filtering, etc. Algorithmic transformation of data (e.g., encoding, encryption)
4. *Interpretation*. (U-Nodes) Fitness, reasoning, rules, etc. Uses semantic description to classify/transform data and reasoning about decisions. Decision, reasoning and rules about interpretive data.
5. *Evolution.* Selects and transforms services according to their fitness to achieve their goals. Decision, reasoning and rules about services according to goals. Fitness reflects the goals description.

The general idea of the BIONETS service architecture is to break down high level user request to low level service invocations. In the proposed logical layers (see

Figure 3). Each layer provides appropriate means to control the layer below. In other words, the evolution layer modifies the interpretation layer over time.

4.1.1 Technical Service Definition

As a first step towards a common agreeable definition on services in the scope of the BIONETS and as a solid ground for the definition of service architecture principles, we worked out the following assumptions:

a) A service is a self-contained component with a well-defined functionality.
b) Services are invoked, configured, and managed through uniform interfaces.
c) Each service comprises a set of operations and parameters, while the operations cover the actual service logic and parameters allow configuring the behaviour of the service logic in a fine grained manner (e.g. timeouts, buffer sizes, number of parallel threads, etc.)
d) T-Nodes and U-Nodes provide the services and allow accessing their interfaces.

4.1.2 Service Types

We distinguish between three types of services, called Fixed Services, Migrating Services, and Composed Services. All services types are interfaced in the same way, but differ in their behaviour and scope.

4.1.2.1 Fixed Services

Fixed Services abstract functionality special to certain nodes, i.e. usually the features of physical devices abstracted by these nodes. For instance a call setup service is supposed to be fixed, since this service can only be executed by a phone device and its execution does not require nested services of other nodes.

4.1.2.2 Migrating Services

The movement of a service execution from one node to another node is supposed to be a matter of state transfer between these nodes. In case, a transfer of service logic between two nodes is also required, e.g. to propagate services, we envision Migrating Services for the moment. Migrating Services are also based on the functionality of a single node, i.e., the features of device abstracted by the node. In contrast to fixed services, the functionalities required for the execution of a Migrating Service are supposed to be provided by each U-Node. Thus, solely the non-functional characteristics of the nodes (e.g. execution speed, power consumption, etc.) may have an impact on the services.

As the name suggests Migrating Services, i.e., the code of the service implementation, as well as the execution state is moved between different nodes on demand of a third party responsible for monitoring and managing the service environment. A possible example of a Migrating Service is a proxy server. The intention of Migrating Services is to have the opportunity to strategically decide which node will provide a certain service. Processing power, storage capacity, network connectivity or even an exposed position in the network topology could be criteria leading to this decision. Migrating services interact with the system executing them in a fashion that is not comprehensible to other services; they are just consuming resources. Thus, the migration of a service can be regarded as a process that comprises the undeployment from one node and deployment to another node.

4.1.2.3 Composed Services

Composed services incorporate various services of multiple nodes to form value added services addressing complex problems. For that purpose the order of utilizing third services is represented in an abstract machine interpretable language which may be processed by each U-Node. The creation of composite services is supposed to be an autonomous process and may also be realized by planning approaches based on evolution. Moreover, composed Services are allowed to include other Composed Services. A composed service is accessible through each U-Node providing a service that is part of the composition.

Since the execution of a composed service is based on services of multiple nodes the execution may also be controlled by all involved nodes cooperatively.

4.2 Service Framework

In principle the BIONETS service architecture bases on the assumption that no certain knowledge about the computing environment's composition is available at any point in time. Consequently, all elements of the service framework are designed to cope with this view. The service framework defines where the different requirements of the architecture are addressed, proposes approaches for service creation, deployment, composition and evolution. Moreover, the framework identifies a number of interfaces which allow accessing the architecture features.

All node interfaces comprised by the service architecture are based on a common interaction framework. The interaction framework is introduced to gain a solid ground for the establishment of exchangeable interaction models for service management and service provisioning. An interaction model determines how data is exchanged between the nodes of the computing environment in order to achieve a concrete interaction semantic. Thus, an interaction model provides specifications for the layers Representation and Processing illustrated in Figure 3. Examples for interaction models are RPC, publish-subscribe, data spaces, blackboards, or information filtering. The definition of an interaction model on top of the interaction framework comprises specifications about data organization, control primitives, and their semantics (e.g., publish, put, etc.), data representation (e.g., XML, binary, comma separated values, etc.), and roles of interacting parties (e.g. publisher, subscriber). For the definition of interaction models within the framework it is assumed that all nodes represent elementary data resources. These resources may be accessed with a limited set of operations for data retrieval and data manipulation. The initial operations are *create*, *read*, *update,* and *delete*. Consequently, all interaction models are supposed to be defined on data exchange driven by these atomic operations.

The interaction framework, i.e., the option to switch between multiple interaction models, is supposed to support the dynamic configuration of the BIONETS system behaviour with regard to the communication.

4.2.1 Runtime

The runtime part of the service framework addresses the service activation and the service execution.

4.2.1.1 Service Activation

Services need to be activated that their operations become accessible. If they are deactivated they do not run any processes and they cannot be executed on demand.

4.2.1.2 Service Execution

According to SOA principles [He03], services are accessed via a uniform, platform-independent interface. This interface is not a classic component interface with an open description, as for instance supported by Web Services or CORBA. Instead, the interface is a generic sink for Service Requests. A Service Request contains a semantic description of the task to be performed (e.g. Inputs, Outputs, Preconditions, and Effects). Since each node only provides a limited set of services, nodes only understand those Service Requests that address services they provide. Thus, Service Requests do not explicitly address a service provider by a name, instead the service features described in the Service Request are supposed to address the required service (operation) and consequently its provider. For that reason, Service Requests are expected to be delegated and filtered in a content-driven fashion on their way to the nodes, but without naming a concrete destination at all, which perfectly fits with the peer-to-peer nature of BINETS communications [BIODelivD111]. Moreover, services are not necessarily executed immediately or their execution may be long-running. Hence, the service response is originated in an asynchronous fashion so that a Service Requests may also be seen as subscriptions for a service response.

4.2.2 Runtime Configuration

A major aspect of the service architecture is the possibility to modify the service environment in order to optimize its behaviour. For that purpose services may be configured and transferred between different nodes.

4.2.2.1 Service Configuration

As suggested above, each service provides a set of configurable parameters. These parameters allow modifying the service behaviour. Consequently, the configuration parameters are supposed to be changed in an evolution-based improvement of the service fitness with regard to resource consumption and processing accuracy of a given task.

Similar to the interface for Service Execution the Service Configuration interface is uniform and independent of service logic and implementation platform. The configuration parameters are changed with a Configuration Request. The representation of Configuration Request is again oriented to reasonable semantic description. The service to be configured is also addressed by its features instead of a certain name. Moreover, Configuration Requests are – on a descriptive level – related to Service Requests to ensure that the right service is configured.

4.2.2.2 Service Migration

For the moment we envision service migration as a special aspect of runtime configuration. Each U-Node that hosts a Migrating Service may be requested to move this service to another node. For that purpose we envision a Migration Request that identifies a particular Migrating Service analogous to the Service Request, but instead of executing the service, the owning node responds with the implementation and the current execution state of this service. The requesting node is consequently supposed to deploy the service, while the node originally hosting this service should undeploy it.

4.2.3 Service Comprehension

Although service execution is supposed to be realized with no or only minimal knowledge about the service environment (i.e. available services), especially planning and optimization processes require information describing node capabilities and services features. For that purpose the classic, descriptive elements for service provisioning as known from UPnP and Web Service are integrated. A simulation approach is intended to address the shortcomings of extensive semantic process description envisioned for today's service architectures.

4.2.3.1 Service Description

The service description explains what a service does, how the input data for the service should be structured and which results are to be expected under the given environment conditions (that is, Inputs, Outputs, Preconditions, and Effects). From that point of view the Service Description is similar to the Service Request, neglecting the fact that the description is provided by the service and not by the service requesting party. Since in classical SOA approaches (e.g. Web Services [BHM+04]) the service description comes with the name of the service provider, its intention in this architecture is an informal one only. Thus, service descriptions are supposed to be originated by the service providers on a general request for available resources for a particular application domain. As a result, the requesting party obtains enough information about the surrounding environment to make a picture of the available services, their features, and probably the capabilities of the nodes providing these services.

4.2.3.2 Extending Service Description with Simulation

A promising way to achieve loose coupling in terms of software component dependencies are heavy-weight semantic service description ontologies like Semantic Web Service Ontology [BBB+05] or Web Service Modelling Ontology [RLK+04]. In fact, in these ontologies the elements used to describe services are based on complex logics, which cause expensive reasoning processes. Especially for simple services the interpretation of the service description may be even more expensive than the actual service execution.

We envision a simulation approach to experience how a service will behave in a certain situation. The simulation is supposed to extend the service description while addressing especially the process-related consideration about the service behaviour. Therefore, each service supports Service Requests flagged as simulation and replies with a description of the expectable result and effects for the given inputs and conditions.

4.2.4 Request Delegation

As suggested in the sections above all interactions are request driven, i.e., the requesting party originates a request that somehow reaches its destination. However, the request does not contain an address determining its destination. The low-end approach to achieve this behaviour is to multicast requests to all listing receivers.

The efficient, spatially decoupled distribution of data among the nodes is a matter of BIONETS WP1.2, but to limit the distribution ways already on the application level we separate service environments based on functional and non-functional aspects (e.g., context information, network topology, etc.) into service spaces. Each service space contains a minor number of nodes and services. The spaces are connected by special delegation/filtering services hosted by border nodes. These delegation services do not answer a service request directly, but forward them to other service spaces, according to external information, e.g., environment context or user preferences. For instance, services that allow controlling and monitoring the devices in a certain room are organized in another service space than internet shopping services. The delegation services are supposed to autonomously form a coherent overlay network and gather arbitrary user and environment information to best accomplish their goal. The following Figure 4 illustrates aforementioned service space.

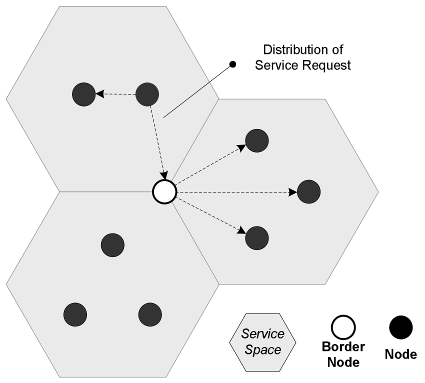

Figure 4: Visualisation of nodes, service spaces and delegation services.

4.2.5 Autonomous Service Composition

An inherent feature of the BIONETS service architecture is the autonomous composition of new services either on demand or in a proactive mode to provide novel services. The creation of Composed Services is supposed to be realized by U-Nodes and involves planning and evaluation mechanisms based on concepts of evolution.

4.2.5.1 Gathering Common Application Domain Knowledge

The creation of a service implementation based on existing services requires general knowledge about certain application domains, i.e. appropriate ontologies, explaining the terminology to describe the concepts of these domains. In general those ontologies should be available in a public data network, but since we expect the computing environment to be partially disconnected from global infrastructures these ontologies should be cached by the nodes and distributed among each other on demand.

4.2.5.2 Gathering Environment Information

The composition of services requires controlled planning and association of available resources. Therefore, a planning phase is initiated with gathering the available resources while asking for appropriate service descriptions. Indeed, the picture of the environment is neither guaranteed to be complete nor long-lasting, but a good starting point for a controlled planning phase. As opposed to classic service discovery the service description does not include an identifier of a service instance.

4.2.5.3 Planning Compositions

The Planning of a service is started with an initial creation of a certain number of first generation plans. Therein, a plan is an abstract program with multiple services as building blocks. These building blocks may for instance be assembled with common programming language concepts as, e.g., conditionals, while loops, forks, or joins.

To evaluate the quality of a plan we envision two different approaches for the moment.

1. All plans are tested in parallel, while using a simulation mode for the nodes and services in the environment. The plans that performed best may be successively improved, e.g., by evolution approaches. If a plan is found that matches the requested service goals this plan is established as executable service. Alternatively a threshold for the quality of service approximation could be introduced if there is no definite solution to expect.

2. With each service execution one plan is tested in life mode. Several plans can also be executed in parallel if several instances of the resulting service are supposed to be executed concurrently. After all plans ran at least once, those that performed best are chosen for successive improvement. Obviously, this approach assumes that the created plans meet the requested service goals.

4.3 User-Driven Service Generation

Services are supposed to come up as solutions for complex user tasks. The description of tasks to be solved as well as the derived service request descriptions are supposed to be represented with proper logics and self-describing syntaxes (e.g. XML). The utilization of proper logics, e.g., in terms of the Semantic Web, reduces the coupling in terminology between different parties, i.e., the comprehension of semantics is not tight to the exact match of vocabularies.

4.3.1 User Request

The User Request contains a formal description of the goal the user wants to achieve with the support of the computing environment. The User Request is supposed to be based on concepts of human thinking instead of concepts on technical solutions.
Thus, an appropriate request may for instance describe the user's desire for light, not the need to switch-on a particular lamp or to lift the curtain. However, a goal description may additionally contain specifications of flows and conditional actions, since these concepts are also immanent to human thinking.

4.3.2 Generic Service Request

The Generic Service Request is the counterpart of the User Request and describes a technical approach to achieve the goal formalized in the User Request. For that purpose the Generic Service Request specifies how several functional elements, i.e., services, need to be arranged for execution in order to fulfil a User Request. Therein, the arranged elements may be regarded as service request themselves. The Generic Service Request is based on technical concepts, indeed, on a semantic level. Additionally, it also incorporates knowledge about the context of the computing environment and the user at the moment the service was requested, assuming this knowledge was relevant to the service functionality. For instance, the user's desire for light usually refers to his or her current location if no other location is explicitly mentioned.

4.3.3 Abstract Service Implementation

The service implementation represents an assignment of currently available service instances to the single elements arranged in the Generic Service Request. Thus, each service implementation may be regarded as one binding for a Generic Service Request. Therein, each element of the request may be realized by exactly one service or again by a composition of services, dependent on the service's complexity. When interpreting the abstract service implementation, the composing elements will be allocated, bound, and scheduled for execution.

4.4 Security

Services are the most flexible component in BIONETS. To create more complex services, they can be put together using existing services. This combination is not limited to a single U-node where services run on but combinations of service instances on different U-nodes are possible too. On top of that services may evolve and thus yield optimised, adapted services which have never been seen before.

To comply with the methodology of services providing functionality to the network, security is also seen as a service. This concept is explained in more detail in [MPS06].

Although the concept of services is pretty much understood it is currently not quite clear how a service and its execution are going to be implemented. Thus it is hard to come up with very precise security requirements.

4.4.1 Security Requirement Description

We assume that a service will not implement all possible security mechanisms foreseen in BIONETS. It will not implement an access control system, it will not provide a complete authentication framework, it will not possess integrated integrity assurance to name just a couple of possible security functionalities a service may use.

On the contrary we imagine that a service is going to use other services to secure its operation and comply with its security requirements. For this to work, each service will need some kind of description which describes the necessary components to be used. In a way this is comparable to a combination of a security policy and a construction manual [Hut06, HV06, IM02].

Evidently, such a description must be secured itself. The binding to a service and the integrity of the description have to be ensured.

4.4.2 Identification

From a security point of view, the identification of a service is a hard issue. To our knowledge there is no identification scheme fixed yet. Therefore, this section explains why such an identification scheme is definitely required.

BIONETS will implement an evolutionary framework and a trust and reputation management system. To assess the performance of a specific service we will need some identification mechanism which allows us to identify it.

In current systems, services are usually identified by the port number they run on. More specifically, when using such a service we use an instance of it which is currently running on a server and bound to a specific port. If the implementation of the service is somehow vulnerable then the corresponding vulnerability reports identify the version of the vulnerable service and the way it is to be fixed. Such an approach is fine if you assume a static environment in which the system administrator knows which versions or packets she has to update in order to remove possible vulnerabilities or to update to newer versions.

However, BIONETS needs much more. We need to take a specific implementation of a service, identify its performance in terms of user satisfaction, fairness, power consumption, reliability, etc. and then assess whether it is worth to keep this service implementation in the network or to remove, modify, or evolve it, as it behaves, for example, maliciously. Thus, assessment will be the first step in such an evaluation framework. It is not an easy task as one has to make sure that the right service is evaluated. An even more critical step is the removal of unwanted services. Such a mechanism will have to reliably find and invalidate the correct service.

Possible identification mechanisms are required to not simply use names or versions of the service. In a disconnected environment this approach will inevitably create ambiguities. A very basic approach may be to use hashes which map service implementations to an ID of the service. However, it may be worth looking in other fingerprinting technologies which provide more information than just *arbitrary numbers*.

4.4.3 Authentication

As services are mainly initiated to accomplish a specific task on behalf of a user, they may also be required to authenticate on behalf of a user, to access for example a database which has restricted access rights. It is clear that this task may be either implemented explicitly in the service itself or it may be provided by a security service [MPS06]. In both cases the service is required to carry the credentials for the authentication procedure.

In another scenario a service may need to be able to authenticate its identity to another service. This could be the case if, for example, a provider sets up a security policy for a service *u* which prescribes that it can only transfer data using a specific service *s*. Service *s* will then have to authenticate to the service *u* distributed by the provider. In order to accomplish authentication, service *s* is required to be involved in the authentication process (e.g. by providing credentials).

In both examples a service may simply carry credentials to prove its authenticity but not the methods to do so. The functionalities can be provided by the U-nodes. However, how the authentication process will work reliably in the absence of a central entity is not solved yet.

4.4.4 Access Control

As mentioned in [MPS06], U-nodes are required to provide an execution framework which isolates services to protect them against each other. The higher level concept of access control will also be required in order to ensure only authorised usage through specified interfaces. To illustrate this requirement we just give an example: A service may try to access another service on behalf of a user which is not allowed to use such a service. Access control prevents this action.

Of course, access control can be implemented implicitly within the service. However, to keep services simple it is proposed to use either special security services which implement an access control system or to use the execution framework to accomplish the access control task. The latter option will require the Security Requirement Description (see above) to contain the appropriate information.

4.4.5 Integrity

If you are able to simply take out a service from BIONETS (e.g. because you have the U-node in your hands), modify the service at will, and inject it in the network afterwards your system might be endangered. Imagine a service which is well known in the network and whose integrity is not protected by, for example, the identification mechanism. You may modify the well known and possibly popular service to have some malicious side effects. Additionally the security requirement description, possibly delivered with the service, may be changed also at will and operations which require special security mechanisms are now performed without them and so revealing arbitrary secrets, offering free access, etc.

Thus integrity protection of services and of the metadata delivered with them is strongly required.

4.4.6 Reputation System

If the identification scheme allows us to find services with a specific characteristic (services which contain a certain malicious logic for example) a reputation system may not be needed. As we currently do not have such a solution a reputation system for services is required.

To illustrate the requirement, assume a service which is evolved in a specific U-node. Now, the service is distributed to other U-nodes. At some point in time the modified service shows malicious actions, or its reputation drops to zero due to unreliable actions. Now the service needs to be removed. We can do this by assigning this revocation task to the U-node which initially initialised the distribution. But without a history functionality implemented in a reputation system, how do we find out, which node created the service through evolution, to which nodes the service was distributed, or whether the malicious service has already been modified again, by, for example, being evolved again. Who is the originating node of this service in the latter case? Another solution may simply ignore all these problems and simply inform other U-nodes that a service with a specific identifier should be eliminated. However, if such an identifier is, for example, simply based on a hash over the source code of a service, then the malicious code may continuously be distributed in the network. The possible effects on security in such a case are not known yet.

4.5 Autonomic Services

In the envisaged BIONETS environment the number of small connected devices (U-nodes and T-nodes) is much higher than current computing systems and all these devices are connected to the some network. Current technologies and service/network architectures will have problems in operating this kind environment. The current service solutions need manual operations in all stages of their life-cycle. They need to be created, deployed and managed manually, therefore needing constant supervision from the human actors in the service chain.

In the BIONETS environment, the number of devices and also the number of services running on those devices exceeds the number of human users by several orders of magnitude, making manual configuration and management unrealistic to implement. The future BIONETS service environment is dynamic, heterogeneous and resource constrained, therefore in order to be able to survive and evolve in such environment without relying on a centralized control BIONETS service needs to be able to automatically manage, optimize, monitor, repair, protect and evolve itself. Therefore in order to cope with this new situation, BIONETS services, as the network, must be truly autonomic.

The aim of this section is to analyse, what kind of autonomous functions are required by the BIONETS services and also to give a short description how these functions could be achieved in the BIONETS computing environment.

4.5.1 Autonomic Functions & Self-Management for BIONETS Services

The increasing number of devices creates possibilities for new applications, but also adds the complexity and dynamics of the networks. The key question is how this complexity and dynamics can be handled in a sensible way without requiring users to become technical experts in the field, and the network owners to spend time and resources in managing dynamic networks [Bett05].

The increasing system complexity is reaching a level beyond human ability to manage and secure. This increasing complexity with a shortage of skilled IT-professionals points towards an inevitable need to automate many of the functions associated with computing today. [IBM06]

One quite obvious and potential solution to these problems would be to increase the degree of autonomicity in the system. The term “autonomic system” can be understood as a computing system that can manage itself given high-level objectives from administrators. One very good example of existing “autonomic system” is the autonomic function of the human central nervous system. Autonomic controls use motor neurons to send indirect messages to organs at a sub-conscious level. These messages regulate temperature, breathing, and heart rate without conscious thought [IBM06]. The BIONETS project has been inspired from this solution and also other similar solutions where “autonomic functions” solves very challenging problem. The aim of the BIONETS project is to have connected, organized and intelligent service components which will give us what we need, when we need it, without any user effort. [IBM03]

According to IBM [IBM06] there are eight defining characteristics of an autonomic system:

1. An autonomic computing system needs to "know itself"
2. An autonomic computing system must configure and reconfigure itself
3. An autonomic computing always looks for ways to optimize its workings
4. An autonomic computing system must be able to recover from routine and extraordinary events.
5. An autonomic computing system must detect, identify and protect itself against various types of attacks.
6. An autonomic computing system must know its environment and the context surrounding its activity
7. An autonomic computing system must function in a heterogeneous world and implement open standards
8. An autonomic computing system must anticipate the optimized resources needed while keeping its complexity hidden

These eight elements from IBM’s vision fits quite nicely also to BIONETS environment. It can be seen that certain basic autonomic elements are needed in the BIONETS services and these “sub-functions” called, self-management, self-optimization/self-configuration, self-monitoring/self-repair and self-protection are presented more detailed in the following sections.

4.5.2 Self-management

The essence of autonomic computing systems is self-management, which intent is to free the users from the details of system operation and maintenance. In the BIONETS environment service self-management implies some support for autonomous decision-making insides the services. These autonomic services will maintain and adjust their operation in the face of changing components, workloads, demands, and external conditions. The management process can configure nodes in the BIONETS environment in a traditional way (by changing values of parameters, etc.) but also provide them management rules that are then executed on the node. The self-management function has to support all the traditional aspects in system management (fault-tolerance, configuration, accounting, performance, and security) and specifically reflect the highly self-organized nature of the BIONETS system, inherent distribution and the continuous changes in the system.

The self-management procedures in an autonomic system can exist in many levels. At first, automated functions can merely collect and aggregate the available information to support decisions by user. Or, procedures can serve as advisors, suggesting possible courses of action for users to consider. In some current systems the autonomic systems can even make some lower-level decisions on behalf of the user. The purpose of the BIONETS system is to take the autonomic procedures even further, where the users will need only to make relatively less frequent predominantly higher-level decisions, which the system will carry out automatically via more numerous, lower level decisions and actions.

When analysing more detailed manner what kind of functionalities self-management in a BIONETS a service needs, the following functionalities emerge:

- *Self-awareness:* An autonomic service should be intelligent enough to be aware of its purpose, and it should have enough information to fulfil its purpose. It will need detailed knowledge of its components, current status, ultimate capacity, and all connections to other systems to govern itself. Because of the disconnected nature of BIONETS environment, each service cannot rely only to the information received from the other services, because they are not necessarily available in the next moment of time. Therefore the service should be automatically situation aware and detect the environment conditions to be able to act as a stand alone situation in a proactive and self-aware way.
- *Self-adaptation:* The environment of the system is continuously under change situations, which require changes in the system internal and external behaviours. The service should automatically detect the meaningful changes happening in the environment, and adapt to the system internal and external behaviour accordingly.
- *Environment-awareness:* An autonomic service cannot exist in a hermetic environment. It must know its environment and the context surrounding its activity, and act accordingly. It should find and generate rules for how best to interact with neighbouring systems.

If we analyse even further we get also specific functionalities which need to be supported by the BIONETS services, e.g.:

- *Service migration*. A Service moves/copies itself automatically to another U-node. This could be also autonomic replication or distribution, but it probably is application dependent choice. This operation needs also fast adaptation functionalities, so that the service can rapidly adapt itself to changing platform and environment.
- *Service creation/deployment:* The BIONETS service doesn't necessarily need to be created/deployed by user, but it can be also automatically created and deployed. The BIONETS system could somehow recognize the needs of the user by using users contextual or history information and see that the user will need specific service.
- *Service composition: A* Service gets the "goal" from the user or detects that some new functionalities are needed and automatically translates these to lower level requirements and finds the services that provide the needed functionality. Services use the fitness values to choose a best available service offering a specific functionality among many similar services.

4.5.3 Self-optimization/self-configuration

An autonomic BIONETS service locates in a changing and heterogeneous environment where there are different parameters and environmental variables to consider, therefore in order to have optimized performance all the time, the service should never settle for the status quo, it should always look for ways to optimize its workings and seek opportunities to improve its own performance and efficiency.

Installing, configuring, and integrating large, complex systems is challenging, time-consuming, and error-prone especially when required to be done dynamically at run time. In BIONETS system the service should configure and reconfigure itself under varying unpredictable conditions in accordance with high-level policies. The service configuration should occur automatically, as well as dynamic adjustments to that configuration to best handle changing environments. This auto-configuration allows the service to adapt itself to existing network connection and sensors.

4.5.4 Self-monitoring/Self-repair

In a large complex system like BIONETS it could take weeks from a team of programmers to find and repair a problem. Therefore the BIONETS service should be capable to monitor itself and try to detect possible software and hardware problems as they appear, and also, if possible, try to repair these problems immediately.

4.5.5 Self-protection

The BIONETS computing environment can be quite dangerous place and despite the existence of firewalls and intrusion detection tools, humans must at present decide how to protect systems from malicious attacks. BIONETS service must be capable of self-protection. It must detect, identify and protect itself against various types of attacks to maintain overall system security and integrity. The BIONETS service should automatically defend against malicious attacks and use early warning to anticipate and prevent system-wide failures. For this purpose a BIONETS service may use or be subject to special security services enforcing for example confidentiality, integrity, or authenticity of data.

4.6 Evolution

The requirements for BIONETS point out the necessity for a BIONET service to be able to dynamically change during its lifetime. In particular, a service must adapt its behaviour to the changing environment and conditions that it is able to sense [PVE06].

Like in biology, we distinguish between *adaptation* and *evolution*. In biology, *adaptation* refers to the capacity of a given organism to sense, respond and adapt to its environment, while *evolution* refers to emergence of new, better adapted species in the long run. In terms of services, *adaptation* is a lightweight mechanism based on a hard-wired closed-loop algorithm that observes the environment and acts accordingly. On the other hand, *service evolution* implies the ability to acquire new functionality not previously engineered into the system [TY05,MYD06].

A simple example is the well-known congestion avoidance mechanism of TCP. A TCP source is able to adapt its current sending rate according to the available bandwidth in the network. However, the algorithm according to which it reacts to the feedback coming from the network (in the form of packet loss events) is static and cannot be modified by the system itself. As such, TCP is able to perform adaptation, but does not support evolution. Evolution is currently supported by humans, typically through researchers and/or programmers that come up with a new version of TCP suitable, say, for a satellite link. An example of self-evolving TCP would be if, in response to sensed network conditions (e.g. a satellite link) a new, more suitable version of TCP, would automatically be created and replace the old one.

In order to be truly autonomic, services must have the capability to evolve in the long run, beyond simple closed-loop short-term adaptations. Otherwise, humans have to cater for these services, performing the upgrades by hand. In other terms, evolution is a “meta-adaptation” mechanism, or a means to produce new adaptation schemes. In literature however, the two terms evolution and adaptation are often confused, since evolution can also be regarded as a form of (long-term) adaptation.

In order to fulfil these requirements, we think that the way the service is defined and implemented must be sufficiently flexible along with the service framework that hosts services.

More concretely, we require the architecture of the service to be able to dynamically change; secondly, we require that the strategy used by the service to conduct these changes pertains to an evolution strategy. Possible strategies to be considered within the BIONETS project may be inspired by genetics evolution, by global behaviour of individuals as observed within human or animal societies, by multi-players game theory, etc. The study and further selection of the most adequate evolution strategies is out of the scope of the research activities conducted in the present Subproject. Indeed, it is the role of activities conducted in Subproject 2, *Methodologies and models for interaction, evolution and dynamics borrowed from the living, social and physical world*. This means that our purpose here is ‘only’ to provide a service architecture and corresponding hosting framework with evolution capability, according to virtually any plugged evolution strategy.

As a cross-fertilisation of both subprojects research activities, in the future we intend to be able to clearly identify, which service architecture features are effectively needed for implementing an evolution strategy. Consequently, as a side-effect, it may happen that some evolution strategies considered adequate from the theoretical point of view will not be suitable for being implemented in the BIONETS service context.

4.6.1 Adaptation of the Service Architecture

Service Architecture Adaptation covers the realization of the evolution of the service architecture that is steered by the evolution strategy. This aspect is necessary for the BIONETS evolution strategies to be implemented.

The functionalities that should be considered are:

- Reconfiguration, i.e. change in the structure of the components that implement the services
- Migration of services
- Removing of services, or unplugging of services
- Creation of a new service, addition/plugging of new components
- Cloning
- Code evolution (could be implemented at first instance by replacement of a component by another one)

Overall, we require services to be implemented following a software architecture that enables dynamic reconfiguration. We can surely get inspiration from modern software components frameworks, such as those proved suited to complex distributed, possibly self-adaptive applications (e.g. large-scale distributed applications like the ones deployed and run on grid computing infrastructures [APPTVVZ05], [ABCCVVZ06], or with self-adaptive capabilities [DL03], [DGPCYP04]).

4.6.2 Evolution Strategies

Evolution Strategy is the central aspect of the BIONETS. A structured approach for implementing evolution strategies would be from rules that define the evolution strategy (e.g. genetic evolution rules), and from a semantics associated with those rules, to generate a set of mechanisms that trigger auto-adaptation of BIONETS services.

Evolution strategies generate mechanisms which, depending on fitness criteria, trigger adaptation of the services [Nag04], [SFHKMRUVV04]. Ideally, evolution should be realized in a totally autonomic way, possibly following a collaborative approach; and we foresee that in most cases the global evolution pattern and resulting decision are distributed. Feedback and fitness measurement, sensing capabilities are aspects that should also be added to the primary functionalities of a BIONETS service.

More precisely, next phases of the research agenda should address the following list of questions:

- Refine the definition of evolution, determining which parts of the services it impacts (Parameters, Code, or both, that for this should be expressed in such a way that they are evolvable), and does it allow interfaces of a service to evolve, or should they be kept immovable.
- Evolution in the state of the art includes: migration, inheritance, creation and injection of code or of new combinations of ready-made modules [YT06]. It may be processed in an incremental way, starting from parameters, going down to pieces of code It may follow biologically inspired mechanisms [Nag04], such as encoding of evolvable properties as “genes” in service “chromosomes”, where genetic operators (mutation, crossover) over chromosomes create new service variants , besides natural selection where only fittest services survive. The population is the result of the services spreading onto the network (U-nodes).
- Service goals can be shared and should be reached in a collaborative way. More precisely, robust selection/fitness/feedback evaluation mechanisms are needed. In particular, feedback may be implicit, e.g. in the sense that popularity of services measures the corresponding service quality. Quality of service may call for effective insurance that the service works correctly (for this, resilience through possible preliminary usage of test-beds. or even dynamically based on redundancy techniques may be used). Care must be taken because feedback may be manipulated intentionally, the current environment may have some more or less fast influence on the feedback. It must also be clear which entities contribute to provide feedback, how and by which entities the feedback is propagated and processed, etc.
- Cascade improvements should be possible, either:
    - Top down: an improvement at a higher-level service requires other improvements at the services that it uses
    - Bottom up: an improvement (could be by chance) at a lower service enables new improvements above it.
- Any change in the system would probably have big security implications, that should be specifically addressed within research conducted in WP4. Also, security requirements, such as trust could probably be mapped into some fitness values for giving the evolution process some security-awareness.

5. Next steps and future refinements

In this deliverable we have presented the initial requirements for BIONETS service architecture rising from the selected application scenarios. We have also presented initial principles for BIONETS service architecture, execution framework and generic description and requirement specification of services in BIONETS SerWork architecture. The purpose of this deliverable was to provide initial framework for BIONETS service architecture research and to illustrate some of the critical service requirements for the research tasks in subprojects SP1 and SP2 as well as to give basic guidelines for SP3 research work.

This document will be further revised and updated based on the research results especially from the WP3.2, and the scope of the deliverable will be enhanced with more detailed service architecture definition. The next revision of this document, D3.1.2, will be available at month M18 from the beginning of the project, providing an intermediate architecture definition and revision of the architecture requirements. The main future refinements include e.g.

- More detailed architecture description using specific UML diagrams.
- Further analysis of application scenarios from the service point of view is needed in order to identify e.g. evolution aspects (this work is carried out in co-operation with SP1).
- Definition of service creation, announcement, and discovery mechanisms in the BIONETS service and network architecture based on the research activities in WP3.2.
- Crystallization and analysis of impact of self-organizing and disappearing network environment for service architecture.
- More detailed definition of evolution from the service point of view including mechanisms with which the services are able to monitor and sense changing environment in order to evolve.
- Crystallization of impact of evolution and possible first algorithm solutions from subproject SP2 to be incorporated with initial service architecture description.

Based on the research results from the SP1, SP2 and SP3, the final BIONETS service architecture definition will be available at the end of the project.

References

[APPTVVZ05] Aldinucci M., Petrocelli A., Pistoletti E., Torquati M., Vanneschi M., Veraldi L., Zoccolo C., ”Dynamic reconfiguration of grid-aware applications in ASSIST” in 11th Intl Euro-Par: Parallel and Distributed Computing, ser. LNCS, vol 3648, aug. 2005.

[ABCCVVZ06] Aldinucci M., Bertolli C., Campa S., Coppola M., Vanneschi M., Veraldi L., Zoccolo C., « Autonomic Grid Components : the GCM Proposal and Self-optimising ASSIST Components », in HPDC'15 workshop on HPC-GECO/Compframe, June 2006. http://www.di.unipi.it/%7Ehpc-geco/GECO-CompFrame06.html

[DL03] David P.C, Ledoux T., "Towards a Framework for Self-adaptive Component-based applications", DAIS 2003, ser. LNCS, vol 2893.

[MYD06] D. Miorandi, L. Yamamoto and P. Dini, "Service Evolution in Bio-Inspired Communication Systems", to appear in Proc. of SOAS06, and in Int. Trans. Syst. Science and Appl., 2006.

[LH04] Liu H., Parashar M., Hairiri S. "A component-based programming framework for Autonomic Applications", 1st IEEE Int. Conf. on Autonomic Computing (ICAC-04), May 2004.

[Hut06] Dieter Hutter. Possibilistic information flow control in MAKS and action refinement. In ETRICS, pages 268-281, 2006.

[HV06] Dieter Hutter and Melanie Volkamer. Information flow control to secure dynamic web service composition. In Proceedings of Security in Pervasive Computing: Third International Conference, SPC 2006, volume 3934 / 2006 of LNCS, pages pp. 196-210. Springer Berlin / Heidelberg, April 2006.

[IM02] IBM and Microsoft. Security in a web services world: A proposed architecture and roadmap. Technical report, IBM amd Microsoft, April 2002.

[MPS06] F. Martinelli, M. Petrocchi, D. Schreckling. Security architecture and infrastructure draft. Technical report, University of Hamburg, April 2006

[Nag04] R. Nagpal. A Catalog of Biologically-Inspired Primitives for Engineering Self-Organization. AAMAS 2003, LNAI 2977, pp 53—62, 2004.

[PVE06] G. Polyzos, C. Ververidis, E. Efstathiou. Service Discovery and Provision for Autonomic Mobile Computing. WAC 2005 Revised Papers, LNCS 3854, 2006, pp 226 -- 236

[DGPCYP04] P. Dini, W. Gentzsch, M. Potts, A. Clemm, M. Yousif, A. Polze. Internet, GRID, Self-adaptability and Beyond: Are We Ready ?, Proceedings of the Database and Expert Systems Applications, 15th International Workshop on (DEXA'04) -. Pages: 782 - 788

[SFHKMRUVV04] G. Serugendo, N. Foukia, S. Hassas, A. Karageorgos, S. Mostefaoui, O. Rana, M. Ulieru, P. Valckenaers, C. Van Aart. Self-organisation: Paradigms and Applications. AAMAS 2003 Ws ESOA, LNAI 2977, pp 1—19, 2004.

[TY05] Christian Tschudin and Lidia Yamamoto: "Self-Evolving Network Software", Praxis der Informationsverarbeitung und Kommunikation (PIK Magazine) 28 (2005) 4, K. G. Saur Verlag, Munich, Germany, December 2005.

[YT06] L. Yamamoto, C. Tschudin Experiments on the Automatic Evolution of Protocols Using Genetic Programming, WAC 2005, LNCS 3854, pp 13—28, 2006.

[IBM03] http://www.research.ibm.com/autonomic/research/papers/AC_Vision_Computer_Jan_2003.pdf

[IBM06] http://www.research.ibm.com/autonomic/overview/elements.html

[Bett05] Bettstetter et al. 2005. Self-organization in Communication Networks. Overview and State of the Art. WWRF White Paper. 44p

[BIODelivD111] BIONETS Deliverable D1.1.1, Application Scenario Analysis, Network Architecture Requirements and High-Level Specifications

[BBB+05] S. Battle, A. Bernstein, H. Boley, B. Grosof, M. Gruninger, R. Hull, M. Kifer, D. Martin, S. McIlraith, D. McGuinness, J. Su, S. Tabet, "Semantic Web Services Ontology (SWSO)," April 2005, http://www.daml.org/services/swsf/1.0/swso/.

[RLK+04] D. Roman, H. Lausen, U. Keller, E. Oren, C. Bussler, M. Kifer, D. Fensel "Web Service Modeling Ontology (WSMO)", D2v1.0, September 2004, http://www.wsmo.org/2004/d2/v1.0/

[He03] H. He. (2003, September). What Is Service-Oriented Architecture. Available: http://www.xml.com/pub/a/ws/2003/09/30/soa.html

[BHM+04] D. Booth, H. Haas, F. McCabe, E. Newcomer, M. Champion, C. Ferris, D. Orchard. (2004, Februrary) .Web Services Architecture. Available: http://www.w3.org/TR/ws-arch/

Terminology

AP-node	Access point to existing network infrastructure
BIONETS	BIOlogically-inspired autonomic NETworks and Services
CORBA	Common Object Request Broker Architecture
Service	A BIONETS architecture entity offering services to users and other services.
Serwork	A higher level architecture containing both BIONETS network and service architectures
SOA	Service Oriented Architecture
T-node	Simple sensor nodes
U-node	Complex nodes offering services to users