Service Evolution in Bio-Inspired Communication Systems

Daniele Miorandi [1] Lidia Yamamoto [2] Paolo Dini [3]

[1] CREATE-NET, v. Solteri 38, 38100 Trento (Italy)
Email: daniele.miorandi@create-net.org
http://www.create-net.org
[2] Computer Science Department, University of Basel
Bernoullistrasse 16, CH-4056 Basel (Switzerland)
Email: Lidia.Yamamoto@unibas.ch
http://cn.cs.unibas.ch/
[3] London School of Economics and Political Science
Houghton Street, WC2A 2AE – London (United Kingdom)
Email: p.dini@lse.ac.uk
http://www.lse.ac.uk/

Abstract: An autonomic network must work unsupervised, therefore must be able to respond to unpredictable situations. The BIONETS project is working towards resilient network services that are able not only to perform short-term adaptations to the environment but also long-term evolution of new functionalities. To this end, a bio-inspired approach is proposed, based on an extension of evolutionary computing to a pervasive environment where disconnected operation is common, and where the fitness of a service is evaluated at runtime. Crossover or recombination of existing services occurs as opportunistic exchange of parameters or code, producing new generations of services which proliferate in the network or are discarded by a mechanism similar to natural selection.

In this paper we review the research lines related to autonomic service evolution currently in progress within BIONETS. A catalytic graph model describes the flow of opportunistic evolutionary interactions, shaped by cascade fitness evaluations. We present a research agenda and possible avenues leading to self-evolving services, and discuss their potential impact on future service engineering.

Keywords: service evolution, bio-inspired models, autonomic communications, evolutionary computing, catalysis, autonomic computing.

1. Introduction

In this work, we address the problems related to the design and deployment of self-evolving services in an autonomic communication system. The scenario we target is that of pervasive computing/communication environments [29], where a myriad of embedded devices with sensing/identifying capabilities provides the necessary means to run situated context-aware services. This is acknowledged to be one of the most challenging scenarios from both the networking and service provisioning point of view, since conventional paradigms do not fit well the peculiarities of such environments. In particular, critical issues related to scalability and complexity (in terms of system management) arise.

The solution we envision builds on biologically-inspired paradigms for the design of autonomic services. The underlying reasoning is that nature has been shown to be able to successfully cope with scale and complexity issues in a fully distributed and uncontrolled manner. The goal is to obtain a framework in which services are able to organize themselves in a purposeful system without requiring any manual intervention. This is in line with current research trends in autonomic computing [24] and autonomic communication [27], which we wish to complement with the distributed on-line evolution of services in pervasive environments.

The large scale of deployment and dynamic nature of such services make it unfeasible to rely on manual software updates. The system should be able to update itself by evolving new functionality automatically.

In this paper we review some of the research lines currently pursued within the framework of the BIONETS project [1, 26]. In particular, we present an extension of evolutionary computing to a distributed disconnected environment, where the fitness of a service is evaluated at runtime. Crossover or recombination of existing services are employed to produce new generations in an asynchronous fashion, which proliferate in the network or are discarded on the basis of their fitness values, leading to a mechanism similar to natural selection.

The remainder of the paper is organized as follows. In Sec. 2 we describe the application scenario and system model we are concerned with. In Sec. 3 we review some related work and discuss some of the issues related to the design of self-evolving services. In Sec. 4 we present a model, based on a catalytic process, for the distributed service evolution framework we propose. In Sec. 5 we present and discuss four possible approaches to service evolution, constituting incremen-

This work has been partially supported by the European Commission under EU project BIONETS (IST-FET-SAC-FP6-027748, www.bionets.org).

tal steps towards the achievement of fully automated services self-organization. Sec. 6 discusses the need of complementing the approach presented with a comprehensive theoretical framework. Sec. 7 concludes the paper presenting some of the research lines we are currently investigating.

2. Autonomic Services in Pervasive Computing Environments

The target scenarios are pervasive computing/communication environments, characterized by a huge number of embedded devices with sensing/identifying capabilities, able to provide the necessary support to context-aware services. These scenarios are characterized by scale and complexity figures which challenge the current approaches to networking and service provisioning/management.

First, in terms of networking it is practically not feasible to build a *fully connected* system. This comes, on one hand, from the unpredictability of channel fluctuations and device mobility patterns, which lead to the necessity of designing a robust system, able to work without taking connectivity for granted. On the other hand, this comes as a natural *design choice*, based on some theoretical work which showed that (i) fully connected networks scale poorly [16] (ii) by giving up the connectivity constraint a scalable network model[1] can be built [15, 12].

The BIONETS system is therefore designed to operate in the absence of a fully connected network. A snapshot of the network would look like an archipelago, where a myriad of islands (each composed by a limited number of connected nodes) will exist. The mobility of (a subset of) the nodes will be used to ensure system-wide spreading of information by means of a suitable opportunistic forwarding scheme [21, 10]. These techniques are based on store–copy–and–forward operations, where a node which gets one message not destined to it stores it and releases a copy of the message to (a subset of) the nodes it encounters, giving rise to a sort of epidemic spreading. In such system, there is a clear tradeoff between the delay that can be tolerated by an application and the level of "connectivity" seen by the application itself (in that over a longer time interval a large number of meetings among nodes takes place).

Second, we need to face the problems related to service provisioning over such large-scale disconnected networks. Conventional centralized (or infrastructure-based, e.g., Web Services) solutions cannot work, since they do not fit well the disconnected architectural assumption of BIONETS. Moreover, aside from the provision of individual services, a system-wide control of the behaviour of the running services would still be needed, a task of unsurmountable complexity. One possible approach to cope with such issues is to resort to *autonomicity* as the paradigm for designing services able to self-organize into a purposeful system. While the self-$*$ properties of autonomic systems [24] would provide the necessary features for obtaining an unsupervised system able to achieve a given desired behaviour, there is no clear consensus in the research community on how to actually design systems able to show these properties. One of the most promising research lines is to draw inspiration from *biology* to introduce novel design guidelines for systems able to show an autonomic behaviour. The reasoning behind such choice is that nature has been successful in dealing with scale and complexity issues, leading to the arising of ecosystems able to self-sustain in the absence of a central control. In particular, *evolution*, understood as the ability to develop new functionalities, represents a key requirements for autonomic services, in order to let them be able to adapt to new, unpredictable, operating situations. The BIONETS project is therefore seeking ways of introducing services able to evolve without requiring manual intervention, while at the same time being resilient to network failures and disconnections.

[1] By scalable network model, we mean that in a network of n nodes, $\frac{n}{2}$ acting as sources and $\frac{n}{2}$ as receivers, the per-connection throughput can scale as $\Theta(1)$, instead of the $\Theta\left(\frac{1}{\sqrt{n}}\right)$ bound in [16], with probability approaching 1 as $n \to +\infty$.

3. Concepts and Background

Our work can be framed within the general context of autonomic communication systems. In particular, a similar bio-inspired approach has been proposed by the group of Prof. T. Suda at UCI, where they designed and developed a biologically-inspired platform for supporting adaptive agent-based distributed services [25, 19, 18]. Our work differs in that (i) we do not limit our scope to agent technology (ii) we account for the possibility of the network to be disconnected, so that the networking aspect is integrated into the evolutionary process.

A related field, in which scientists are facing scale and complexity issues similar to the ones we are considering, is that of large-scale computing systems (e.g., peer-to-peer computing, grid computing etc.). Researchers in the area have investigated the applicability of bio-inspired models for engineering large-scale computing systems, and have shown that, in a large variety of situations, they may lead to solutions which outperform conventional approaches [5].

3.1. Biological Evolution and Evolutionary Computing

One of the simplest yet effective processes in biological systems is evolution. Evolution refers, roughly speaking, to a measurable change in the features of a population over successive generations. Evolution is currently understood as an outcome of a process of natural selection, where individuals with a higher level of fitness (i.e., with more favourable features for the surrounding environment) are more likely to survive and reproduce.

This is a brief explanation of what biologists mean by evolution. On the other hand, evolution is a concept which has been used for a long time in computer science. One of the reasons for that is the simplicity of evolutionary processes. They rely on the existence of a population of individual entities which can give rise to a new generation by means of two mechanisms, which can be easily implemented as algorithms

in a computer: transformations (such as crossover and mutation) and selection. This is at the basis of Genetic Algorithms (GAs), where the natural selection process is used to build an iterative process able to find approximate/optimal solutions to optimization and search problems. Similarly, the same concept has been applied to find computer programs able to best perform a given task, giving rise to the branch of Genetic Programming (GP).

GP and GAs follow basically the same steps:

(i) **Representing candidate solutions:** Candidate solutions to a proposed problem are modelled as a population of individuals. Each individual (candidate solution) is encoded using a genetic (genome-like) representation or genotype. The genotype usually takes the form of a bit stream for GAs and a program tree for GPs.

(ii) **Fitness evaluation:** Each generated candidate solution is rated by means of a *fitness* function that evaluates the suitability of the solution to solve the proposed problem.

(iii) **Selection:** A *selection* mechanism operates on the result of the fitness evaluation to select a subset of solutions to "survive" to the next generation.

(iv) **Generating new candidate solutions:** One or more *genetic operators* (mutation and crossover being the most common) modify the genome to generate new candidate solutions. A *mutation* is a modification of a small portion of the genotype chosen at random. *Crossover* consists in randomly selecting genotype segments in two individuals and swapping these segments between them. The result are two new individuals which are inserted in the population.

(v) **Iterative search and optimization** The generation of solutions, evaluation and selection process continues in an iterative way until a solution with maximum fitness is found, or until the fitness of the generated solutions remains stable or shows little improvement over generations. The solution with maximum fitness is then taken as the output of the algorithm.

GAs and GP are usually performed as off-line tasks in a centralized manner. This does not fit the nature of BIONETS services, where the evolution process is fully distributed and needs to be performed in real time. In order to solve this problem, the first step is to extend the GA/GP steps above to a distributed context. In section 4.2 we show our initial proposal for such an extension. Although there are other efforts to apply GAs/GP in a distributed context [8, 25], these are either synchronous extensions of GA/GP [8] to support parallel computing, or do not cope with disconnected operations and low-level network services [25].

3.2. Towards Self-Evolving Services

We now turn our focus to the meaning of *service evolution* in a communication system. We distinguish between *adaptation*, intended as the ability of a given system to tune its behaviour according to the current environment, and evolution. Adaptation is a "light" mechanism, in that it follows a hard-wired adaptive (closed-loop) algorithm. On the other hand, by evolution we mean the ability to acquire new functionality not previously engineered into the system. A simple example is the well-known congestion avoidance mechanism of TCP [20]. A TCP source is able to adapt its current sending rate according to the available resources in the network. However, the mechanism according to which it reacts to the feedback coming from the network (in the sense of packet loss events) is *static* and cannot be modified. As such, TCP is able to perform adaptation, but does not support evolution.

It is also worth noting that evolution can be considered at both the *macro* as well as *micro* level. By micro level, we mean the level of individual entities (in this case, individual services). By macro level, we mean the system-wide level, in which different services coexist, cooperate and compete as in a digital ecosystem [2]. The two scales have different dynamics and require different tools to be modelled and analyzed. At the micro scale, we deal, e.g., with self-tuning of some running parameters up to self-generation of code. At the macro level we deal, e.g., with replication of successfully services, Nash equilibria for shared resources among competing services etc. The link between the micro and the macro level is provided by interactions. Interactions may help in progressing evolution at the micro level (by spreading successful services) and are at the basis of evolution at the macro level. Nonetheless, since we are dealing with a large-scale complex system, where a large number of mutually interacting dynamical entities coexist, the study of the micro level is not sufficient for understanding/predicting the behaviour on a system-wide scale.

3.3. Phases in Service Evolution

The service evolution process can be divided into two distinct phases: bootstrapping and adaptation. Bootstrapping refers to the operations involved in the creation of a service from a set of requirements. From a bio-inspired perspective, we can make an analogy to the "origin of life" problem, in which self-sustaining complex living structures emerged out of interactions between simple molecules. During the service bootstrap phase, an initial version of the desired system would be constructed out of basic building blocks, given some high-level objectives. The second phase consists in a sort of incremental process, where the service is adjusted to maximize its fitness, i.e. its ability to achieve the target goals.

The bootstrapping phase can therefore be understood in terms of the self-generation of a service, where the ability to self-organize (i.e., create order out of chaos) makes it possible for evolution to take place. Obviously, this is an extremely difficult and challenging problem with no generic solution in the short term. However, as discussed in Sec. 3.4 below, autocatalytic models used in origin of life hypotheses offer us a useful interaction model which can be applied to the full evolutionary cycle.

The second phase is more realistic and feasible in the short term: it boils down to optimizing an existing, working system that had been previously engineered by humans. This phase can be better understood in terms of control systems, and will be dealt with in Section 3.5.

3.4. Autocatalytic Networks and the Bootstrap Phase

Researchers studying the origin of life problem have come up with hypotheses based on chemical reaction models, on how life would have emerged out of simple molecules. In this context, a simplified system of chemical reactions is studied in [22]. The problem is to understand under which conditions order (in the form of structure and complexity) may appear given an initial set of simple molecular species which interact in a pseudo-random fashion.

A chemical reaction takes a number of *substrates* or *reactants* and produces a number of *products*, sometimes with the help of a *catalyst*, a substance that accelerates or facilitates the reaction without being consumed in the process. A molecule may act as catalyst or inhibitor for different reactions. When the supply of substrate is abundant with respect to the amount of catalyst, the rate of product formation depends directly upon the amount of catalyst. The model in [22] relies on this simplifying assumption, and models molecules as catalysts or inhibitors for the production of other molecules. Moreover they consider the system evolution over time-scales long enough so that any molecule may interact with any other molecule in the system. The least fit molecules (in terms of relative population) are dropped at each round and are replaced by new molecules.

While perhaps too simplistic from a chemistry point of view, this model can be easily transposed to the service evolution case: services can be modelled as molecules that act as catalysts for other services, providing incentives for them to spread over the network, or may inhibit services that are not suitable.

In [22] catalytic and inhibitive features are modelled by means of a directed graph, where a positive (respectively: negative) weight associated to a link $i \rightarrow j$ means that molecule i is a catalyst (respectively: inhibitor) for molecule j. No self-loops are considered.[2] It is shown that such model experiences a phase transition, consisting in the sudden appearance of a set of molecules that take over the others, i.e. some form of "order" emerges. The arising of order can be well understood here by considering autocatalytic sets (ACSs). An autocatalytic set is defined as a set of molecular species which contains a catalyst for each of its member species. ACSs act as self-replicating structures, even if none of the constituent molecules can self-replicate in isolation. Two examples of ACSs are shown in Fig. 1. In this figure, the nodes s_i are catalysts according to the simplified model in [22], and the links between nodes have a constant weight $c_{i,j}$ representing the strength of the catalytic (when $c_{i,j} > 0$) or inhibitory (when $c_{i,j} < 0$) effect (only a few sample node labels and weights are indicated, for the sake of simplicity). The simplest ACSs are cycles (also referred to as hypercycles [13, 4]), but not all ACSs are cycles; on the other hand, every ACS contains at least one cycle. It is possible also to show that (i) ACSs appear with high probability in a finite time, given any initial condition (ii) once an ACS appears, the phase transition takes place at exponential speed [23]. This example shows the importance of interactions (and cooperation) in evolution.

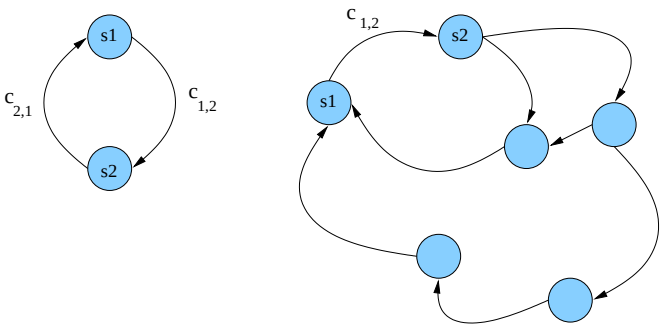

Figure 1: Two examples of autocatalytic sets.

It is worth adding a remark with respect to the absence of self-loops in the model. Self-loops represent a degenerate form of ACS, in that a self-loop with positive weight would indeed characterise, in the simplified chemical framework, a molecule able to self-replicate. The dynamics of such molecule is of little interest (the concentration would just grow exponentially over time). It is however not clear at the moment if self-loops can provide an interesting model for service evolution, i.e., whether services should be allowed to self-replicate (like worms do). It is interesting to note that self-loops correspond to a quasispecies model in systems biology, describing the dynamics of a system of self-replicating entities. This has proved to be useful in modelling the processes of self-replication of macromolecules (DNA, RNA). Self-replication can provide diversity by means of mutations (taking place as errors in the process of replication) and is used also as a model for asexual reproduction of bacteria and viruses. Its direct applicability to service evolution is an open issue, due to obvious security considerations. This is an example of the typical trade-off between security and flexibility (provided here by automated functionality extension).

3.5. Adaptation Phase: Feedback and Fitness

One of the critical issues for evolutionary processes to take place is the implementation, by means of an adequate fitness function, of a natural selection process. Let us focus on what we termed the second phase of evolution, namely the situation in which we already have a running service that needs to be adapted to changes in the environment.

The first question is: what do we mean by "environment"? In a communication system framework, it is clear that the environment should include the available system resources, as well as the user's satisfaction level (at least for functional services). The fitness can be therefore thought as consisting of two components: (i) an objective component (user-independent), which reflects the properties of the running service in terms of performance, i.e., ability to attain the target goals (ii) a subjective component (user-dependent) which reflects the level of satisfaction of the end user. While part (i) can be implemented in a software system (it just requires a

[2] The absence of self-loops ensures that no self-replicating molecules can be present.

monitoring unit), part (ii) seems more challenging to obtain. Indeed, it implies a lot of work in terms of human-computer interface, for understanding, in a transparent automated way, what the user expects from the system.

Assuming that somehow this "fitness" level can be estimated, the next step is to determine how to react to it. In general, the fitness can be understood as a feedback signal, generated by the environment. The general picture is therefore that of a classical closed-loop control system, as in Fig. 2, acknowledged to represent a general model for self-managing systems [11]. The system is constituted of the service unit which receives an input from a unit (called evolution engine) which, upon evaluation of the service fitness (by considering the actions taken, comparing them with the high-level goals and considering also user-dependent fitness components) determines the modification to be done to the service itself. The inner feedback loop in Fig. 2 accounts for the fact that a service, as discussed above, can include a hardwired adaptive mechanism. Such inner loop is able to change only the behaviour, but not the functionalities of the service. The evolution engine comes into action when functionality changes are involved.

Feedback constitutes a fundamental ingredient for evolution. Feedback in the form of fitness is the driver of the natural selection process.

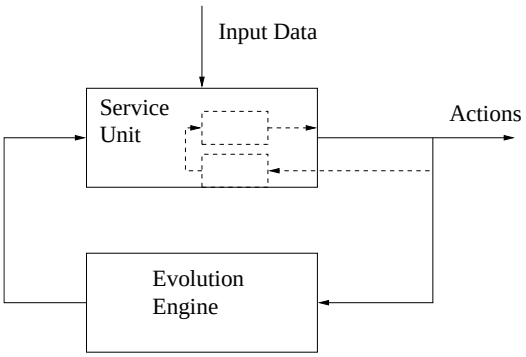

Figure 2: Representation of a self-evolving service as a feedback control system. The dashed inner-loop control represents an adaptive mechanism hardwired in the service code.

4. Service Evolution Model

In this section, we present a catalytic model for service evolution in a dynamic disconnected environment. The model is based on an extension of evolutionary computing approaches, where fitness is evaluated at runtime in a distributed fashion. Crossover or recombination of existing services exploits opportunistic contacts between services, producing new generations of services whose diffusion in the system is controlled by a mechanism similar to natural selection.

4.1. Evolution and Interactions

In evolutionary biology, it has long been understood that interactions play a fundamental role in evolution. The simplest example is sexual reproduction, where interactions between individuals result in increasing genetic diversity of the offspring without relying on potentially harmful mutations. It is therefore a primary need to understand what kind of interactions we can have between services in the BIONETS setting.

We are dealing with a distributed communication system, in which interactions (understood as the possibility of transmitting information between service entities) take place by means of communication. Therefore, the ability to communicate is a *necessary* (but not sufficient) condition for the ability to interact.

A service in BIONETS is defined as a software entity able to perform some well-defined task for a given client or set of clients. Clients may be end users, higher-level applications, or other service entities. The task is well defined in the sense that clients possess the means to periodically evaluate how well the task has been performed, obtaining a fitness value. Services are then selected or discarded based on their fitness values. [3]

In order to perform the task assigned to it, a service may need to deploy modules or components in several places in the network. A service is therefore a distributed entity. Typical examples are communication-related services, such as a reliable transmission protocol (requiring at least one source and one sink at different locations) and a multimedia conferencing service (requiring multiple media modules located at each user end point). The fitness of such distributed service must somehow reflect its overall performance, and is therefore more difficult to obtain. A distributed service should be robust to failures of its constituent parts, such that the impact of localized failures on the overall fitness is minimized. A model for such overall distributed service evaluation is left for future work.

We start by considering *atomic services*, i.e. services that are composed of a single element whose fitness is evaluated at a single location. In this context, modelling a distributed service may be simplified by locally evaluating each of its constituent parts independently. This model is deficient since it loses sight of the global service behaviour. However it is useful as a starting point.

We model atomic service interactions using graphs, as follows: Given a system consisting of N nodes, each one representing a service, we can define an undirected graph $\mathcal{G}_{interactions}$ having vertex set $\mathcal{V} = \{1, \ldots, N\}$ and an edge between node i and node j if and only if atomic services i and j actually interact.

There are several types of interaction, including client-server, direct peer interaction (for information sharing between peers belonging to the same distributed service), and breeding (crossover) interactions for evolutionary purposes.

An interaction is considered over a given timescale T. Considering the mobility of users/devices/services in the network, we can also construct a connectivity graph $\mathcal{G}_{connectivity}$ over the timescale T, where an edge is drawn between node i and

[3] The platform might also include conventional, static (non-evolving) services, but these are not taken into account in this paper.

j if, over the considered timescale, they happen to belong to the same island of connected nodes, and there exists an underlying bidirectional communication path connecting i and j. If all the nodes were static, $\mathcal{G}_{connectivity}$ would represent, for any T, the overlay topology of connectivity among nodes hosting service interconnection.

To complete the picture, we also need to define two other graphs. The first one is called *evolution graph* and denoted by $\mathcal{G}_{evolution}$, where the existence of an edge (i, j) means that the interaction between services i and j corresponds to the exchange of information for evolutionary purposes (i.e., they can give rise to new services by crossover operations). Services i and j must be alternative implementations of the same service. It is easy to understand that

$$\mathcal{G}_{evolution} \subseteq \mathcal{G}_{interactions} \subseteq \mathcal{G}_{connectivity}.$$

The second graph is called the client-server graph and denoted by $\mathcal{G}_{clientserver}$. It is a *directed* graph, where the existence of a $i \rightarrow j$ link means that service i is a client of service j. As will be discussed in Sec. 4.2, a weight on link $i \rightarrow j$ indicates the fitness of service j as evaluated by its client i. A client-server relationship between two nodes is a form of interaction, therefore:

$$\mathcal{G}_{clientserver} \subseteq \mathcal{G}_{interactions} \subseteq \mathcal{G}_{connectivity}$$

Note that the first inclusion means that the set of directed edges of $\mathcal{G}_{clientserver}$ is a subset of the set of undirected edges of $\mathcal{G}_{interactions}$. This can be achieved either by ignoring the direction of the links in $\mathcal{G}_{clientserver}$, or by replacing every undirected link in $\mathcal{G}_{interactions}$ with two corresponding directed links. Further, it is worth stressing that $\mathcal{G}_{clientserver}$ and $\mathcal{G}_{evolution}$ have no direct relationship with each other, apart from the fact they are subgraphs of the same graph $\mathcal{G}_{interactions}$.

Since the interactions related to evolution rely on connectivity, the network dynamics is one of the drivers of the evolution process. However, as already stressed, this provides just a necessary condition for interactions to take place. We believe that socio-economic processes shall be accounted for in order to reach a complete picture of the interaction process between services in BIONETS. The field of Social Network Analysis [28] will help in the analysis of the interdependencies between the technical infrastructure, the business and service transactions, and the social networks of users.

While the concept of connectivity graph is well-understood in the case of static fixed networks (and a lot of papers within the complex systems community have revealed the implications arising from the properties of the Internet's topology [3]), there is no global theoretical framework for understanding connectivity in a mobile disconnected scenario. In this section we provide a first example on how to define such connectivity graph. For the sake of clarity, we will refer to (atomic) services as nodes of the graph; when we say that two services meet, we mean that the devices hosting them get within mutual communication range and are (physically) able to exchange information. Consider, again, a set of N users, possibly mobile, and fix a time window T. We will focus on the time window $[t_0, t_0 + T)$, for an arbitrarily chosen time instant t_0. Consider the contact process $Z = (t, \sigma)$, where $t_0 \leq t < t_0 + T$ is the time instant at which a meeting takes place[4] and σ is a mark of the form $(i, j),\ i, j \in \{1, \ldots, N\}$ being the IDs of the nodes that become able to communicate at time t. When an island gets formed, all the possible pairs of nodes are considered. At t_0, the process registers all the paths in the initial configuration. We can build a connectivity graph in the following way. Consider an initially empty graph with vertex set $\{1, \ldots, N\}$. If the process Z admits a mark (i, j), we draw an edge between i and j, if not already present. In this way, we can construct the graph $\mathcal{G}_{connectivity}$.

4.2. Catalytic Service Evolution

Given the graph model described in Sec. 4.1, we now propose a model for service evolution inspired by the catalytic reaction model described in Sec. 3.4 and incorporating fitness evaluation as a feedback signal as described in Sec. 3.5.

Recall that the graph of interactions $\mathcal{G}_{interactions}$ represents the set of services that interact in various ways over a timescale T. The two main types of interactions considered in this section are client-server relationships (with fitness evaluation) represented by $\mathcal{G}_{clientserver}$, and the generation of new service variants by crossover, represented by $\mathcal{G}_{evolution}$.

We now extend the algorithmic steps of GAs and GP (Section 3.1) to this graph context, using the catalysis graph model explained in section 3.4 and the fitness feedback model of section 3.5.

Step (i): Representing Candidate Services

Recall from section 3.1 that in GA/GP we generally a set of candidate solutions for a single given problem, while in our case we have a full graph of distinct services, each solving a separate problem as requested by their clients. Furthermore, several instances of the same service may be available in the network, according the demand for that service. In order to model that, we take inspiration from the catalysis model from Sec. 3.4.

Initially ($t = t_0$), a set of candidate services are available, represented as the nodes of $\mathcal{G}_{clientserver}(t_0)$. Each node i has an associated "concentration" value k_i, representing the initial amount of catalyst i available in the system. Fig. 3 shows two examples of client-server interaction graphs, where s_i represents a service and a weight $c_{i,j}$ on a link represents the fitness of s_j as evaluated by s_i. Note that, in constrast with the autocatalytic sets mentioned in Section 3.4 (compare with Fig. 1), we focus on acyclic graphs. We do not know yet whether cyclic graphs would emerge or make a sense in real-world client-server interactions. This remains to be verified in future experiments.

Assuming that the concentrations of reactants are large enough (as in [22]), the rate of product formation when catalyst i catalyses product j is:

[4] Referring to [9] for a more formal definition of the meeting process, we define a meeting as the event that two nodes are able to transmit data.

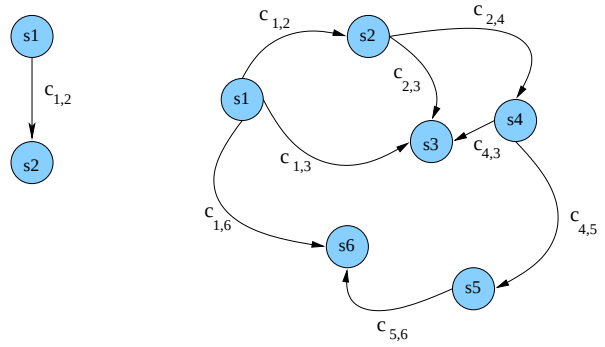

Figure 3: Two examples of acyclic client-server interactions

$$\frac{\partial k_j}{\partial t} = \alpha_{i,j} k_i - \beta k_j \tag{1}$$

where $\alpha_{i,j}$ is a constant weight on link (i, j), and $\beta > 0$ represents a natural "death rate".

The rate of product formation can be controlled by the quantity of catalyst and the constant α, while the rate of destruction of the same product is determined by β.

Considering that one service may have several clients, (1) can be extended to a set of equations for the system dynamics:

$$\frac{\partial k_j}{\partial t} = \sum_i \alpha_{i,j} k_i, \quad j \in \mathcal{V}, \tag{2}$$

where $\alpha_{j,j} = -\beta$ by definition and $\alpha_{i,j} = 0$ if no (i, j) link is present in the client/server graph.

Since the physical location of services is not yet modelled, a high concentration of a substrate means that more copies of the service are available in the network, no matter at which physical nodes these copies are located. In case they are all in the same machine, it actually means that the weight or popularity of this service on this machine is high (the physical process itself is not necessarily replicated, as in the case of a web server serving multiple clients). This model matches well with multiset rewriting and chemical programming, in which copies of the same instance are represented by a simple multiplicity counter [7].

Step(ii): Distributed Fitness Evaluation

In contrast to GA/GP, it is now up to each client to evaluate the fitness of the services it uses. Each service may be used by a set of clients, and may thus receive different evaluation feedback from each one.

In order to model this distributed fitness feedback flow, we define $\alpha_{i,j}$ in Equations (1) and (2) as the fitness of service j as evaluated by its client i during period T. In this way, the amount of a service increases with its fitness and the amount of clients demanding that service. A constant death rate ensures that a service that is not very useful will not indefinitely consume resources in the system.

The range of fitness values must be the same over the whole system. If we adopt the convention of [22] (catalysis versus inhibition) we could establish a fitness range of $[-1; 1]$ with -1 meaning a totally harmful service, 0 a useless service (but not harmful) and $+1$ a totally fit (perfect) service.

Since the weights directly influence the rate of catalysis of the service according to (1) and (2), a negative weight (or fitness) contributes to decrease the service "concentration". When the concentration reaches zero the service is eliminated from the system. This provides a way to eliminate harmful services from the system. When a service i is eliminated, all the services that it used will stop receiving the positive feedback on the fitness, and will eventually also be eliminated.

Step (iii): Selection

The selection mechanism operates according to a chemical reaction model based on multisets [7]: substances that are present at a higher concentration in a chemical solution are more likely to meet other compatible substances with which they can react, or which can catalyse new substances. When looking for a suitable service, a client may send a service discovery query to the neighborhood to find suitable candidates. More popular services are more likely to be hit by such queries, and therefore bind to the potential clients, creating a client-server link in the interaction graph. The service discovery mechanism is orthogonal to the chemical model but influenced by it, in the sense that services occurring at higher concentrations have a higher likelihood to be discovered than those occurring at low concentrations.

Step (iv): Generating new candidate services

The graph system is dynamic. The elimination, addition and movement of services, as well as interaction changes, are reflected by transformations on the graphs $\mathcal{G}_{clientserver}$ and $\mathcal{G}_{evolution}$ which occur every period T, with a direct impact on $\mathcal{G}_{interactions}$. Nodes and links may be added or deleted, and weights may be updated based on new fitness evaluation values.

Two genetic operators are modelled: mutation and crossover. A mutation of a given service s into its variant s' means that at the next instance of the graph ($t_{u+1} = t_u + T$) node s will be replaced by node s'. The potentially new behaviour of node s' due to the mutation might have an impact on its fitness evaluation, therefore the weights on its incoming/outgoing links might change as a consequence.

A crossover between two services s and r at time t_u is represented by an edge (s, r) on $\mathcal{G}_{evolution}(t_u)$. At t_{u+1} the edge is removed and two new nodes s' and r' are inserted, which are the result of the crossover operation. The new nodes have the same set of links in the interactions graph as the original nodes, but their weights may change due to the new fitness evaluation. It is worth remarking that a crossover may lead to an increase in population, since the offspring do not necessarily replace their parents.

Step (v): Distributed continuous optimization

In order to enable continuous evolution, the steps of fitness evaluation, transformation through genetic operators, and selection must be performed at regular intervals, such that new, potentially more suitable services can be generated, discovered and used instead of old ones. The system elements are then constantly being evaluated and modified to adapt to new situations. As opposed to traditional GAs and GP, there is no stop condition, as the system runs continuously.

5. An Incremental Approach Towards Self-Evolving Services

In the previous sections, we have discussed some of the main issues related to the deployment of self-evolving services in large-scale disconnected networks. This is rooted in the recognition of the ability to evolve in an unsupervised manner as one of the key facets of autonomicity in pervasive computing/communication environments. We have discussed various bio-inspired paradigms which represent promising research directions to pursue for designing novel approaches able to overcome the limitations of current engineering practice.

Within the framework of the BIONETS project, we have identified four main steps to be accomplished, in an incremental fashion, for achieving the final target of designing a mechanism able to support the service lifecycle in a unsupervised and efficient way. These steps are graphically depicted in Fig. 4 in terms of potential impact/performance enhancements versus time and complexity, and can be described as follows:

(i) An approach based on genetic algorithms for the optimization and adaptation of selected parameters in atomic services at runtime in a fully distributed fashion, according to the steps in Sec. 4.2. The set of parameters associated to each (atomic) service represents the genome. When two devices meet, they can decide to exchange some genetic material, giving rise to a new generation of services. The convergence properties of such mechanisms have been preliminarily analyzed in [17].

(ii) An approach based on GA-like techniques for the optimal combination of service building blocks. This reflects the current trends toward the design of Lego-like components, which can be assembled at runtime to achieve the peculiar user's goals. Here evolution takes the form of self-assembly. This task will build on the knowledge and software developed within the framework of the DBE project [2].

(iii) An approach based on genetic programming techniques, for the creation of the glue logic that would allow automatic protocol and service composition in such a way to reduce syntactic compatibility between protocol or service modules. Also, GP-like techniques could be used for the generation of the actual code within each protocol and service. Initial experiments on the evolution of simple protocol implementations have been reported in [30].

(iv) An approach which tries to mimic gene expression mechanisms for the automated generation of code. This aims at transferring into code the ontogenic/morphogenesis mechanisms of DNA, in order to try to overcome the limitations (in terms of convergence speed) of all phylogenic GA/GP approaches. Solutions of this kind appear extremely interesting for applications in pervasive computing environments, in that they offer the opportunity of integrating in a seamless way the response of the environment. Another advantage offered by such solution is that it would allow to automatically compose services at the semantic (DNA) level, letting gene expression-like mechanisms take care of the functional specifications.

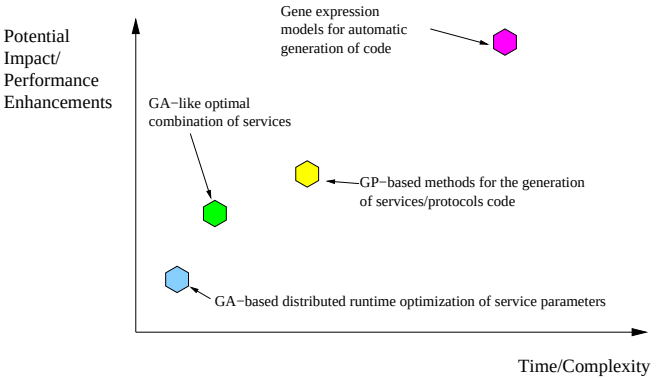

Figure 4: Plot of potential impact/performance enhancements versus time/complexity for the four approaches to build self-evolving services.

6. A Complementary View: Self-Organization and Software Engineering

The BIONETS way to obtain autonomic services builds on a bio-inspired approach, based on crossover or recombination of existing services to produce new generations of services which proliferate in the network or are discarded by a mechanism similar to natural selection. In the previous sections, we have analyzed, from an engineering perspective, some of the issues involved with such approach, depicting an engineering roadmap towards service evolution. Nonetheless, we are aware of the fact that this approach may be short-sighted and may quickly show its limits if not well-complemented by a solid theoretical framework for service evolution.

Generally speaking, we could argue that engineering is gradually approaching the entirely "bottom-up" processes that in biology rely on the cooperation and integration of evolutionary and gene expression dynamics. (This is more evident in software, but a constant process of abstraction is also taking place in more traditional forms of engineering, motivated and justified by the ever-greater reliance on the digital processing of information.)

In Fig. 5 compares the different processes taking place for constructing order (intended as achieving a purposeful system) in (conventional) engineering, software engineering and biological systems. Traditionally, engineering has relied on an

iterative process, where a blueprint is used to produce artifacts and the feedback is used to modify the blueprint. Software engineering has introduced a different process, in which high-level specifications (e.g., UML) are mapped in a top-down fashion into code. Code gives rise, through compilation and instantiation to the actual program. Such a process is context-insensitive and represents, basically, a linear transformation. Feedback from the instance can be used to either fine-tune the UML specifications or, by means of automated GA/GP approaches, be used to derive a better code. Biological systems, on the other hands, work in a quite different fashion. The equivalent of code in software engineering is the genome, which maps to the genotype through an interpretation process. The genome determines the phenotype (the equivalent of the program instance) in a highly non-linear, complex and environment-dependent way (by means of mechanisms such as development, morphogenesis and gene expression).

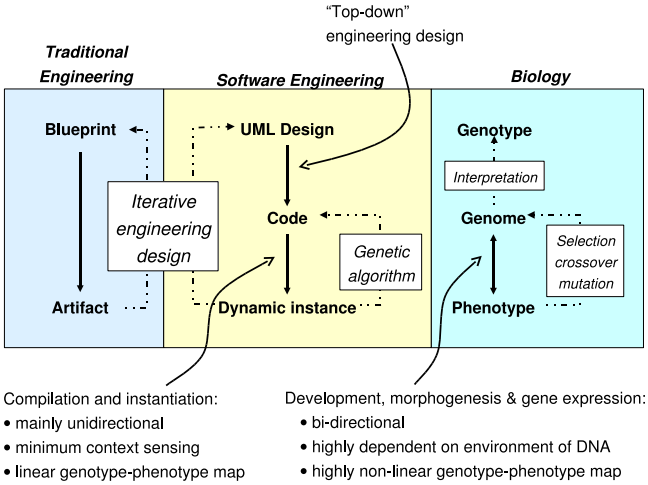

Figure 5: Comparison of different processes for the construction of order.

Fig. 5 is also useful for driving home the point that currently evolutionary computing is largely limited to iterative algorithms, which are conceptually not very different to iterative engineering design. In other words, evolutionary computing can be (and usually is) regarded as a framework for parameterized optimization methods that can be tuned in various ways depending on the problem at hand. In such sense, evolutionary computing alone does not provide a satisfactory theoretical framework that could help in answering key questions such as: How to make sure that services evolved during runtime are feasible, i.e. that they do not result in malfunctioning or malicious services? How to set up a reliable on-line selection mechanism that would effectively eliminate unsuitable service variants, while promoting the most adapted ones?

The deep reason behind this limitation is that evolutionary computing deals mostly with algorithms in isolation. On the other hand, as we have stressed in the previous sections, evolution (and gene expression in particular) builds on the concept of interactions. A model for representing interactions is proposed in [14]. We are currently investigating whether this interactive computing model, together with process algebras such as π-calculus, and chemical models such as γ-calculus [6], could represent a viable direction for modelling service evolution.

7. Conclusions

In this paper, we have presented some of the research lines currently under investigation within the framework of the BIONETS project, focusing on bio-inspired approaches for building autonomic services. The goal is to obtain a framework in which services are autonomic in the sense that they are able to organize themselves in a purposeful system without requiring any manual intervention. We have discussed why evolution, understood as the ability of a system to upgrade its own functionality beyond simple adaptation, represents a fundamental building block of autonomic systems.

We have discussed some of the issues related to the design of a system able to support service evolution, and proposed an extension of genetic algorithms/genetic programming to distributed pervasive environments with disconnected operation. In such model, the fitness of a service is evaluated at runtime by its clients and crossover/recombination relies on opportunistic communications.

We have also identified and discussed four main steps (ordered in terms of time/complexity/potential impact) to achieve the long-term goal of automated service evolution.

We are currently pursuing two main research lines. From an engineering perspective, we are investigating programming models and techniques to build protocols and algorithms able to support service evolution. From scientific perspective, we are working towards the construction of a theoretical framework for modelling evolutionary systems.

Acknowledgements

This work has been partially supported by the European Commission under EU project BIONETS (IST-FET-SAC-FP6-027748, [1]). The authors would also like to acknowledge IST projects DBE (contract number 507953, [2]) and OPAALS (n. 034824). The authors also gratefully acknowledge the contribution of Gerard Briscoe to Fig. 5.

References

[1] BIONETS Project. http://www.bionets.org.

[2] Digital Business Ecosystem (DBE) Project. http://www.digital-ecosystem.org.

[3] R. Albert and A. L. Barabási. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74:47–97, Jan. 2002.

[4] C. S.-O. Attolini. *From Molecular Systems to Simple Cells: A Study of the Genotype-Phenotype Map*. PhD dissertation, University of Vienna, Austria, 2005.

[5] O. Babaoglu, G. Canright, A. Deutsch, G. D. Caro, F. Ducatelle, L. Gambardella, N. Ganguly, M. Jelasity, R. Montemanni, A. Montresor, and T. Urnes. Design patterns from biology for distributed computing. *ACM Trans. on Autonomous and Adaptive Syst.*, 2006. In press.

[6] J.-P. Banâtre, P. Fradet, and Y. Radenac. A Generalized Higher-Order Chemical Computation Model with Infinite and Hybrid Multisets. In *1st International Workshop on New Developments in Computational Models (DCM'05)*, pages 5–14, 2005. To appear in ENTCS (Elsevier).

[7] J.-P. Banâtre and D. L. Métayer. Programming by Multiset Transformation. *CACM*, 1993.

[8] T. C. Belding. The Distributed Genetic Algorithm Revisited. In *Proc. Sixth International Conference on Genetic Algorithms*, San Francisco, CA, 1995. Morgan Kaufmann.

[9] I. Carreras, D. Miorandi, and I. Chlamtac. A framework for opportunistic forwarding in disconnected networks. In *Proc. of MOBIQUITOUS*, San Jose, CA, 2006.

[10] A. Chaintreau, P. Jui, J. Crowcroft, C. Diot, R. Gass, and J. Scott. Impact of human mobility on the design of opportunistic forwarding algorithms. In *Proc. of IEEE INFOCOM*, Barcelona, ES, 2006.

[11] Y. Diao, J. L. Hellerstein, S. Parekh, R. Griffith, G. E. Kaiser, and D. Phung. A control theory foundation for self-managing computing systems. *IEEE Journal on Selected Areas in Communications (JSAC)*, 23(12):2213–2222, Dec. 2005.

[12] O. Dousse, M. Franceschetti, and P. Thiran. On the throughput scaling of wireless relay networks. *IEEE Trans. Inf. Th. - IEEE/ACM Trans. on Netw., joint issue*, 2006. In press.

[13] M. Eigen and P. Schuster. *The Hypercycle: A Principle of Natural Self-Organization*. Springer, Berlin, 1979.

[14] D. Golding and P. Wegner. The Church-Turing Thesis: Breaking the Myth. In *Computing in Europe (CiE) Conference*, 2005.

[15] M. Grossglauser and D. Tse. Mobility increases the capacity of ad hoc wireless networks. *IEEE/ACM Trans. on Netw.*, 10(4):477–486, Aug. 2002.

[16] P. Gupta and P. R. Kumar. The capacity of wireless networks. *IEEE Trans. on Inf. Th.*, 46(2):388–404, Mar. 2000.

[17] I. Carreras, F. De Pellegrini, D. Miorandi and H. Woesner. Service evolution in a nomadic wireless environment. In *Proc. of WAC*, Athens, 2005.

[18] T. Itao, T. Nakamura, M. Matsuo, T. Suda, and T. Aoyama. Adaptive creation of network applications in the Jack-in-the-Net architecture. In *Proc. of Networking*, Pisa, Italy, 2002.

[19] T. Itao, T. Nakamura, M. Matsuo, T. Suda, and T. Aoyama. Service emergence based on relationship among self-organizing entities. In *Proc. of SAINT*, Nara,Japan, 2002.

[20] V. Jacobson and M. J. Karels. Congestion avoidance and control. In *Proc. of ACM SIGCOMM*, Stanford, CA, 1988.

[21] S. Jain, K. Fall, and R. Patra. Routing in a delay tolerant network. In *Proc. of ACM SIGCOMM*, Portland, OR, 2004.

[22] S. Jain and S. Krishna. A model for the emergence of cooperation, interdependence and structure in evolving networks. *Proc. of Nat. Acad. of Scien. of Unit. Stat. of Amer. (PNAS)*, 98:543–547, Jan. 2001.

[23] S. Jain and S. Krishna. Graph theory and the evolution of autocatalytic networks. In S. Bornholdt and H. G. Schuster, editors, *Handbook of Graphs and Networks*. J. Wiley and Sons, New York, NY, 2004.

[24] J. O. Kephart and D. M. Chess. The vision of autonomic computing. *IEEE Comp. Mag.*, 36(1):41–50, Jan. 2003.

[25] T. Nakano and T. Suda. Self-organizing network services with evolutionary adaptation. *IEEE Transactions on Neural Networks*, 16(5), Sept. 2005.

[26] F. Sestini. Situated and autonomic communication: an EC FET European initiative. *ACM Comp. Comm. Rev.*, 36:17–20, Apr. 2006.

[27] M. Smirnov. Autonomic communication: research agenda for a new communication paradigm, 2004. White Paper.

[28] S. Wasserman and K. Faust. *Social Network Analysis: Methods and Applications*. Cambridge University Press, 1994.

[29] M. Weiser. The computer for the 21^{st} century. *ACM Mob. Comput. Commun. Rev.*, 3(3):3–11, 1999.

[30] L. Yamamoto and C. Tschudin. Experiments on the Automatic Evolution of Protocols using Genetic Programming. In *Proc. 2nd Workshop on Autonomic Communication (WAC)*, Athens, Greece, Oct. 2005.

Authors' Biographies

Daniele Miorandi received a PhD degree in Telecommunication Engineering from University of Padova, Italy, in 2005. In 2003/04 he spent one year of his doctoral thesis visiting the MAESTRO project at INRIA Sophia Antipolis (France). In 2004 he had a research appointment with IEIIT-CNR, Torino (Italy). Since Jan. 2005 he is a researcher at CREATE-NET, Trento (Italy). His research interests include design and analysis of bio-inspired communication paradigms for pervasive computing environments, analysis of TCP performance over wireless/satellite networks, scaling laws for large-scale information systems, protocols and architectures for wireless mesh networks.

Lidia Yamamoto is a post-doctoral researcher at the Computer Science Department, University of Basel, Switzerland. Her research currently focuses on autonomic communication and bio-inspired networking. She holds a doctoral degree from the University of Liege, Belgium (2003), a master's degree in Communication Systems and Networks from the Technical University of Madrid, Spain (1995), and a bachelor degree in Computer Science from the State University of Campinas, Brazil (1991). She also worked for Hitachi Europe Sophia Antipolis Laboratory, France (2002-2004), KPN Research, the Netherlands (1996-1998), and Telefonica I+D, Spain (1993-1996).

Paolo Dini was born and raised in Italy and completed his university education in the US: BS Aeronautical Engineering UC Davis (1983), MS (1987) and PhD (1990) Aerospace Engineering Penn State. He taught physics at Carleton and St Olaf Colleges 1992-97 while developing aerodynamics codes for wind turbine aerofoil design, and worked as project manager for ViA Inc, a wearable computer company, 1997-2000. Since 2000 he has moved back to Europe where he has worked at Philips Research Laboratories UK and at the MIT Media Lab Europe in Dublin. Since 2003 he is a Senior Research Fellow in the Media and Communications Department at LSE.