

# Evaluation of two P2P-Approaches for a Distributed Simulation Framework

<sup>1</sup>Christoph Adelberger (christoph.adelberger@fh-salzburg.ac.at), <sup>1</sup>Thomas Kurz (thomas.kurz@fh-salzburg.ac.at), <sup>1</sup>Raimund Eder (raimund.eder@fh-salzburg.ac.at), <sup>1</sup>Thomas Heistracher (thomas.heistracher@fh-salzburg.ac.at)  
<sup>1</sup>Salzburg University of Applied Sciences  
ITS Information Technology & Systems Management  
Urstein Sued 1, A-5412 Puch/Salzburg, Austria  
Tel: +43 50 2211 0, Fax: +43 50 2211 1099

**Abstract.** The paper at hand describes the architecture and functionality of a distributed simulation and testing environment that is intended to simulate the interaction behaviour of Small and Medium-sized Enterprises (SMEs) based on biological similes, e.g. the mechanism of evolution and the structure of the peripheral nervous system in the human body. The simulation framework consists of functional compartments and is distributed in a P2P infrastructure that is built on top of two open-source projects, the ServENT and Soapod. This approach targets at a better performance and additional testing capabilities for the infrastructure. Furthermore, this paper contributes to the open-source projects of choice by evaluating, testing and implementing various services.

**Keywords:** Digital Business Ecosystem, Evolutionary Environment, Agent-based simulation, P2P

## 1 Introduction

The usage of ICT by SMEs evolved from basic communication, e.g. via Web sites, e-mail up to e-business tools and environments of networked organisations. Still SMEs lack innovation, know-how and financial resources to compete with globally acting companies. This development triggered an initiative for convincing SMEs to better adopt ICT usage and to form a collaboration framework, the so-called *Digital Business Ecosystem* – DBE (DBE, 2008). At the moment the initiative is further developed within the OPAALS Network of Excellence (OPAALS, 2008)

The concept of *Business Ecosystem* emerged from the idea of forming an economic community which is supported by a foundation of interacting organisations and individuals. These organisations and individuals represent the organism of the business world. The economic community produces goods and services of value to customers, who themselves are members of the ecosystem. By adapting this concept to state-of-the-art technologies, bringing P2P-based software technologies for transport, connections and searches for services, and enabling the distribution of all stakeholders within this infrastructure, the term *Digital* can precede *Business Ecosystem*.

The DBE is defined as an open-source, free and Internet-based environment in which business applications can be developed and used. It enables end-users to easily access and uses those applications and to have the benefits of intelligence, interaction and adaption as the software evolves from their own usage. Consequently, it forms a network emerging from social connections and a framework of collaboration for SMEs.

The Salzburg University of Applied Sciences (SUAS) contributed in the research area of DBEs since the beginning of the European Union's project DBE. SUAS designed and implemented the *Evolutionary Environment Simulator* – EvESim (EvESim, 2008) in order to run simulations on a biologically inspired P2P environment, a DBE respectively. In general, the EvESim acted as a framework for understanding, visualising and presenting the DBE concepts to contributors within and outside the DBE community, thus constituting a kind of collaboration platform for interdisciplinary research.

Nevertheless, the EvESim faced several challenges. For instance, the simulator was designed and implemented as a standalone application, based on a rich agent-based simulation toolkit. This led to (i) performance issues while simulating complex cases. Furthermore, the DBE bases on highly decentralized and self-organised networks. This implies the issue of (ii) emulating real P2P network behaviour and (iii) adaptively simulating evolving networks of individuals. These challenges led to the idea of redesigning the architecture of the EvESim and distributing the simulator on a P2P network based on the same concepts as the DBE itself. This network is established by two open-source service containers, namely the ServENT and Soapod.

In the following chapters of this work, the architecture of the new generation of the EvESim is introduced, related benefits and limitations are examined and some first results yielded with this distributed simulation framework are discussed.

## 2 Evolutionary Environment Simulator

The *Evolutionary Environment Simulator* (EvESim) provides a simulation framework for biologically inspired P2P systems. It aims at simulating and demonstrating the behaviour of DBEs and also at providing a collaboration framework for the research community. In general it acts as a framework to investigate, present and visualise the idea of DBEs.

In order to analyse non-linear and adaptive interactions which are mostly too complex to be captured by analytical expressions, computer simulations are used. The idea of such computer-aided simulations is to specify and consequently study the rules of behaviour for individual entities as well as their interaction. The simulation itself aims at simulating a multitude of individual entities using a computer model and to explore the consequences of the given rules. As the individual entities can be considered as agents, the simulation of their behaviour and interaction is known as an agent-based simulation.

The Recursive Porous Agent Simulation Toolkit – REPASt (Repast, 2008) is a free and open-source agent-based modelling and simulation toolkit, acting as the framework of choice of the EvESim. It provides a variety of useful functions in order to simulate complex models. These include pre-implemented agents, event schedulers, built-in simulation result logging and graphing tools, libraries for neural networks, genetic algorithms and more.

Although REPASt provides a broad agent-based simulation toolkit for the EvESim it had to be extended to the needs of the DBE. But even the adapted model led to a lack of performance on the one hand, and revealed an overhead of unused features on the other hand. Consequently, the EvESim was separated from the REPASt framework and developed as a standalone application.

The EvESim is intended to simulate the concepts of a DBE, consequently it simulates the behaviour of a collaborative network of SMEs. Depending on the context and logical viewpoint, the terms SME, agent, node and actor are used interchangeably. Each agent within the EvESim provides and consumes services to and from contributors or other agents, respectively. Although, services can be differentiated in general between software services, such as an online-booking service, and real-world services, such as a car rental service, the service model of the EvESim considers services of SMEs as a list of attribute-value pairs: for example, an attribute-value pair represents the price of a hotel room and its corresponding numeric value, such as € 120,-. Moreover, each service defines a service name, a short description of the service, a reference to its provider agent and a migration history. This migration history stores all agents where the service was used and therefore provides the possibility to retrace the service's lifecycle.

The agents of the EvESim simulate the activities of SMEs. Each agent defines a name, a description and several behavioural instructions and settings named social variables. Furthermore, each agent provides a local service pool, a list of services on offer (portfolio), and a list of services on demand (see Figure 1).

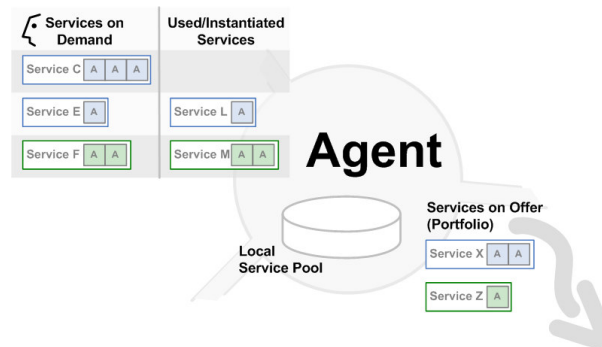


Figure 1: Representation of an agent within the EvESim

The *local service pool* is equivalent to a local service repository. The agent states its clear intentions and interests in the network, and at the same time the system tracks the agent's service consumption. The local service pool data is used to set up the environmental conditions of the services, which migrate to the agent. Consequently, services which are probably useful for an agent will reside at its local service pool. This also provides information about the service distribution within the DBE.

In the *services on offer* (portfolio) the agent manages a list of services which are offered to other agents. When a new service is offered to the DBE, it is defined in the services on offer and simply a reference to it is copied to the local service pool of the producer agent. From there, the services meta-

description or, in other words, reference is distributed to all known neighbours of the agent and consequently, it is advertised to the DBE.

The *services on demand* can also be seen as a profile of the corresponding agent. In other words it represents a list of services that are valuable for the agent but which cannot be produced by it. This service list also defines the interest or ‘intentions’ of the agent. Depending on the simulation’s scheduler, the agent gets the authorisation to request one of its services on demand and looks up the local service pool for adequate services or service compositions. This look up, composition and evaluation are supported by biologically inspired algorithms.

The simulation procedure can be seen as a sequence of tasks, which each agent executes in given time slots defined by a scheduler. The intention of the agent is to find services on offer within the DBE that match its expectations defined in the services on demand. Therefore, triggered by the agent’s request, a biologically inspired optimisation algorithm creates new service combinations and returns them as a possible solution to the agent (Kurz, T. & Heistracher T., 2007). The agent evaluates this solution and if appropriate, it is used as a result.

Even though the development and integration of a simulation framework for DBE concepts was successful and partners from different research domains successfully integrated their findings, constant modifications led to limitations of the framework, such as lack in performance and adaptability problems. Furthermore, the emulation of real P2P networks and the simulation of highly self-organised and evolving networks of agents caused implementation problems. Thereby, this triggered a basic redesign, the encapsulation of the EvESim and its distribution on a decentralised P2P infrastructure.

The concrete problems of the old EvESim architecture were outlined in (Kurz, T., Noguera, J., Eder, E. & Heistracher T., 2007) together with a draft of a distributed architecture. Namely the limitations were:

- the complex model underlying model,
- Repast as the all-in-one bases of the simulation,
- a lack of simulation performance, and
- difficulties in estimating the behaviour of a real P2P network.

During the implementation of the new EvESim architecture as outlined in (Kurz, T., Noguera, J., Eder, E. & Heistracher T., 2007), some modifications had to be included in order to run it fully P2P. The modified and implemented architecture of the EvESim will be described in the following chapter.

### 3 Architecture

In order to overcome the limitations of the EvESim and to provide an adapted simulation framework, the EvESim was redesigned and encapsulated into several basic entities. These entities were distributed on a P2P network, which is built on the same concepts as the *Digital Business Ecosystem* (DBE). This enables the possibility to share simulation resources and to simulate network traffic and network connections in a more realistic manner.

The EvESim consists of several basic functional units, such as a simulation control unit, an agent container unit, a biologically inspired optimisation unit and a reporting unit. Consequently, it seemed obvious to extract those units and encapsulate them into services and form a service-oriented framework. These services are listed in Table 1.

Table 1. Services in the service-oriented framework

• CoreSim	• Status Report
• Agent Pool	• Central Service Pool
• Agent	• Maps
• Genetic Algorithm	• GUI

The following subsections describe these logical entities in more detail.

#### CoreSim

Representing the first service, the CoreSim is intended to be the only centralised service in the

simulation framework. It represents the single entry point of a simulation and provides configuration and evaluation capabilities via a Graphical User Interface (GUI). This GUI is used to configure the behaviour of the agents on the one hand, and to control the simulation itself on the other hand. Furthermore, the CoreSim is responsible for initialising the agents, such as creating services and setting up parameters based on the settings given by the user. After deploying the agents in the agent pools, the CoreSim collects the simulation data and proceeds with evaluation and serialisation tasks.

### **Agent Pool**

The second service is defined as the agent pool. The agent pool represents a container of agents in the simulation. Depending on the configuration an agent pool can have several agents and a simulation can have several agent pools. The agent pool defines the environment for the agents and provides interfaces for communication and interaction with other agents, and the DBE respectively.

### **Agent**

The agent defines and manages the activities of a Small and Medium-sized Enterprise (SME) and represents the main actor in the simulation. Although, each agent could have been implemented as a service as well, it was decided to aggregate several agents in an agent pool in order to reduce the number of services in the simulation network.

### **Genetic Algorithm**

The third service is named Genetic Algorithm (GA) and implements the biologically inspired optimisation component of the simulation framework. Based on a common interface, the network can provide several different implementations of GAs. The GA service is called directly by an agent when it gets the authorisation to request a service from the DBE. GAs are search algorithms, which are based on the mechanics of natural selection and natural genetics. In general, GAs can be split up into four operations: (i) selection, (ii) reproduction, (iii) crossover, and (iv) mutation (Goldberg, D.E., 1989).

With regard to the EvESim, the GA service can be explained by a very simplified example: Considering a hotel room service, the example is extended by a hotel which wants to offer a holidays weekend service. This service offers a hotel room and a candle light dinner on Saturday night. Obviously, the hotel can only offer the hotel room and hence it has to define a service on demand which states the need for a candle light dinner. A restaurant offers a candle light dinner service. Within the DBE, the GA is used to produce the combination of the hotel room service and the candle light dinner service. This service combination is returned to the hotel which wants to offer a holidays weekend service. Consequently, it offers a service combination of separate available services on the network.

### **Status Report**

The fourth service is the status report, which represents the communication channel between the agents in the network and the CoreSim service. As outlined before, the CoreSim provides functionality to evaluate and present simulation data and therefore, it is dependent on a service which provides up-to-date simulation data. However, each agent manages its own data and is responsible for delivering the data to the status report. This collected simulation data can be retrieved and analysed by the CoreSim. Consequently, the status report acts as a push-and-pull queue for storing information.

### **Central Service Pool**

An additional service represents the central service pool. This service pool provides a central service container for all available services on offer in the P2P network, or the DBE respectively. As the DBE is highly decentralised, this service is not part of the DBE architecture but was necessary for the simulation framework in order to analyse the behaviour of the decentralised service reproduction and service distribution.

### **Maps**

The visualisation of the whole network of agents and the underlying infrastructure is implemented in the service named maps. This service provides visualisation components based on Google Maps and Asynchronous JavaScript and XML (AJAX). It is capable of visualising different networks, which evolve during a simulation, such as social networks, producer-consumer networks etc., and provides a user interface to retrieve up-to-date simulation data. These networks are described in an XML tree and

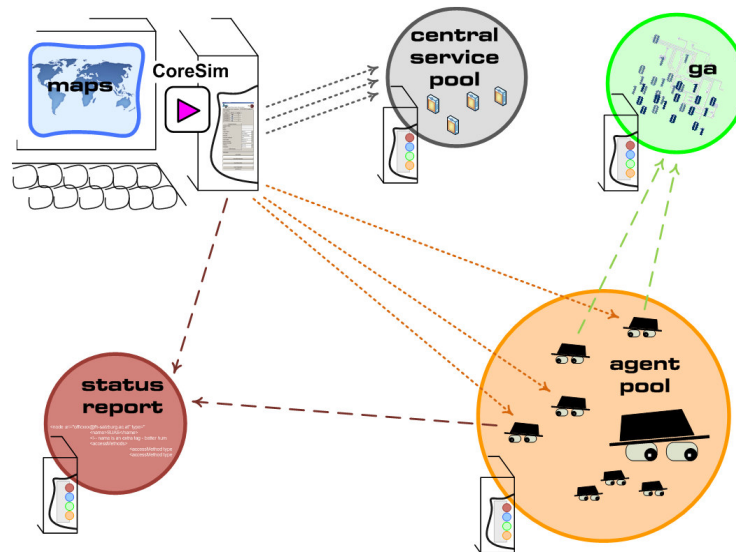
is read by an AJAX application and visualised by Google Maps (Google Maps, 2008). The XML description can also be used for serialisation or other visualisation of the network.

### Graphical User Interface

The Graphical User Interface (GUI) is attached to the CoreSim and represents the configuration and visualisation entity of the EvESim. It is used to configure and control simulation scenarios.

All these services, also called core services of the EvESim, are distributed on the P2P infrastructure and consequently, form a *distributed and scalable simulation framework without centralized authorities*.

The P2P infrastructure is built on top of service-based P2P network nodes, which act as a network of dynamically connected application servers. These application servers are suited for the use within a DBE and rely on the concept of such DBEs. Consequently, they provide an environment to offer and consume services. Figure 2 shows the set up of the distributed EvESim architecture.



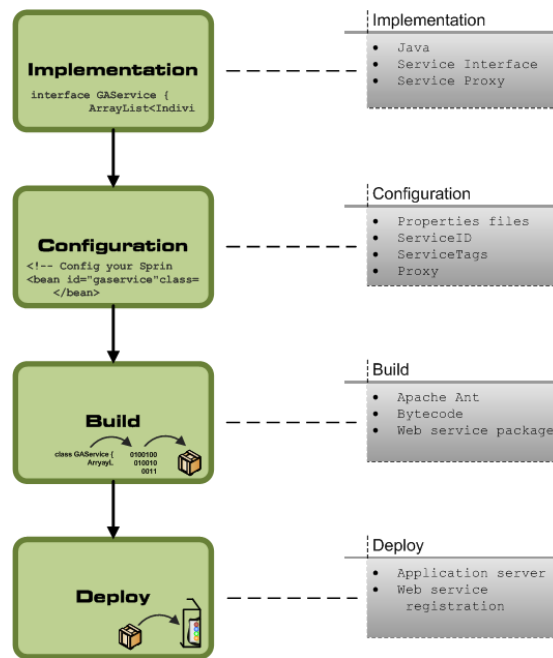
**Figure 2: Core services of the distributed EvESim**

Figure 2 disregards one component, namely the Graphical User Interface (GUI), as it is attached to the CoreSim. Within this environment the EvESim distributes its resources and simulates use case scenarios on top of a DBE-like infrastructure. This enriches the results of the simulation framework.

## 4 P2P Simulation Infrastructure

A main goal of the DBE is to provide a highly distributed and scalable network without any centralized authorities. Consequently, the Execution Environment (ExE) of the DBE (Swallow, 2008) should provide the technological foundations for a fully adapted P2P environment. Furthermore, the shortcomings of the EvESim forced a distribution of the simulation framework.

In order to i) overcome the performance issues and ii) not need to estimate the network behaviour of the simulated nodes, the EvESim framework was implemented on top of the P2P system, developed in the DBE project. So called ServENTs have the functionalities of both, clients and servers in parallel and act as the application servers for the DBE and moreover, also for the services of the EvESim. The advantage beside the increasing power of a distributed system is the fact, that the network traffic of the real system has not to be estimated because the data of the simulation is also distributed on the same underlying P2P system. In Figure 3, the main steps for creating a service on the ServENT are drafted. In order to have a comparison of different technologies, a second P2P approach, the so called Soapod (Soapod, 2008) was used in parallel for the implementation of the EvESim services. By implementing an Abstract Factory design pattern (Gamma, E. et al. 1995), it can be decided before running a simulation if the services should run on ServENT or Soapod. Nevertheless, the respective application server (ServENT or Soapod) has to be connected to the EvESim GUI and the EvESim services have to be deployed at the nodes before a simulation is started.



**Figure 3: Main tasks of creating a service for the ServENT or the Soapod (Adelberger, C., 2008)**

In the following the differences of the implementation on the two P2P systems is described on the basis of Implementation, Configuration, Build and Deploy. Furthermore, some other relevant issues of P2P systems like for example lookup are analysed in the context of the new EvESim architecture. This comparison can be seen as one of the major outcomes of the work at hand, because the two systems were not analysed in such a broad extend beforehand. (Adelberger, C., 2008)

### Implementation

As regards the implementation of the services, each service defines a service interface and a service adapter. While the Soapod service does not need any further implementation, the ServENT service implements an additional interface, namely the *CoreAdapter*, or *Adapter* for non-core services respectively. Besides, this enables the ServENT to distinguish between basic services and core services and consequently, it provides different access to ServENT information. Core services therefore, have direct access to the ServENT's internal methods. The Soapod service is based on open, well adopted Web technologies and standards. Thus, the implementation is able to use Java JSR 181 annotations, which enables advanced configuration of Web services. However, annotations were not used in the EvESim.

### Configuration

As far as the configuration of the services is concerned, ServENT and Soapod are configured differently. Whereas the ServENT relies on a single properties file, in which basic information, such as a service ID, a service name, a service tag list, a service proxy, and a service adapter is configured, the Soapod configuration consists of three separate XML files. Although the XML files can be generated by an Apache Maven plugin, and can be used for any further Soapod service, the XML files provide some obstacles in configuration, due to limited documentation. The configuration requires knowledge in Web standards, Java Servlets and Java Beans. Additionally, the aegis XML binding of Soapod, which is a proprietary binding of XFire, causes difficulties as it is not well documented at the XFire Web site. Nevertheless, XML binding can be changed to other, more sophisticated binding technologies, such as JAXB 2.0 or Castor. But this requires further configuration in the XML files of a Soapod service.

### Build

The service build process was achieved by Apache Ant and therefore, it required slightly different

Ant `build.xml` files. Nevertheless, the only difference was to adapt the `build.xml` files to the appropriate directory structure of the associated file archive (`.dar` and `.war` files). Apache Ant provides the possibility to build the services within the complete EvESim project and extract the required Java source code from it. The Soapod provides an additional Apache Maven plugin to create the project skeleton and all its configuration files. Although this is convenient for single projects, it is not feasible for the EvESim, due to its complex architecture. With Apache Maven the building process of a Soapod service would be much less complex, as Apache Maven provides Web service generation functionality.

## Deploy

Deployment of ServENT and Soapod services is based on Web technologies. As the ServENT provides a HTML POST statement to copy the `.dar` file to the ServENT, the Soapod is based on the Web management toolkit of Apache Tomcat, which provides an administration user interface to deploy or undeploy services. Both, ServENT and Soapod service deploy can be embedded into Java source code. Additionally, ServENT core services provide deploy and undeploy functionality and the Soapod provides an extra service, namely the AdminService, which implements the same functionality. However, these services were not tested in the EvESim.

## Service Lookup

As the services are distributed on the P2P network of ServENTs and Soapods, it is not ensured that each service is available on each node. Therefore, ServENT and Soapod implement different types of service lookup mechanisms. The ServENT is implemented on top of FADA, which enables a distributed service directory. FADA provides a service lookup that is based on a flooding algorithm. Basically, a service is found by sending the service request to a ServENT (Initiator) within the P2P network. If this ServENT is not the service's provider, it broadcasts the query to potential neighbours. The neighbours continue with the same procedure and return the lookup results to the Initiator ServENT. However, this approach consumes high bandwidth and computing power because many nodes of the network are involved. The Soapod implements another strategy to lookup services on the P2P network: Distributed Hash Tables (DHTs). DHTs enable the possibility to search for services without using broadcasts in the P2P network. Soapod uses the `Overlay Weaver` project, which is an implementation of such hash tables. This type of service lookup provides a scalable and efficient mechanism for distributed service access. In order to create the underlying P2P network of ServENTs and Soapods, they need to be connected. This is done by a mechanism named neighbouring.

## Neighbouring

The ServENT and the Soapod implement a neighbour look up mechanism, which is capable of looking for nodes in the network. The ServENT uses an automatic neighbouring mechanism, which connects all available ServENTs. However, this is restricted to a Local Area Network (LAN) due to the use of network broadcasts. In contrast to this, the Soapod provides a configuration file for neighbouring. It holds a static list of host addresses which the Soapod is connected to. This enables it to establish neighbour connections over Internet links.

## Client

Both, the ServENT and the Soapod provide a client toolkit that supports the user in searching for services and instantiating the proxies. The ServENT client toolkit provides a `ClientHelper`, which allows searching for services by IDs and by tags. Additionally, the `ClientHelper` instantiates the service proxy and thus, the ServENT client toolkit returns a fully functional service proxy as answer to a service request. However, in order to generate the proxy, the service's Java interface definition must be available at the client side. In contrast to this, Soapod's services are advertised as Web Service Description Language (WSDL) descriptions and therefore, the Soapod client toolkit returns only the entry points for its services. Thus, the client does not provide the service's Java interface as it is generated from the WSDL description. Even though this procedure is common for Web services and enables distribution at a global scale, the service's proxy generation is more complex than it is with the ServENT service.

## Additional Comparison

During the execution process, both, ServENT and Soapod caused difficulties. First of all, the

ServENT caused classloader troubles while executing a ServENT service. This is due to different classloaders in the ServENT: Each deployed service has its own classloader in addition to a common classloader, which is responsible for global class loading. As different services may contain identical classes, ServENT classloaders caused problems that the classes could not be loaded. The issue was bypassed by packaging all Java classes of all services into a single `.jar` file and loading it with the common ServENT classloader. The single services were reduced to their configuration files. Thus, the common classloader managed all class loading. However, this does not provide an appropriate solution as it requires full access to the ServENT. Secondly, the ServENT caused troubles in the use of resources: Simulating a network of agents causes a Java stack overflow, due to a lack of memory. It was back traced to connected service proxies, which are no longer in use, but were still not closed. Both, the ServENT and the Soapod caused troubles in concurrent modifications: As the simulation is highly distributed and each agent is responsible for its own tasks and behaviour, the simulation framework triggers a high rate of information exchange. This implies the use of shared resources, such as the status report service or the GA service. These services are simultaneously used by several agents on the P2P network and therefore, the resources are changed at the same time. Obviously, ServENT and Soapod do not take care of this and therefore, this has to be taken care of within the Java implementation classes of the services.

## 5 Conclusions and Outlook

The European Commission included the research area of Digital Business Ecosystems (DBEs) in its 6th Framework Program (FP6) and consequently, provided the foundation for improving and enriching the underlying concepts. DBEs aim at providing Information and Communications Technology (ICT) applications and services that improve Small and Medium-sized Enterprise (SME)'s business efficiency within European Union territories. Within the FP6 two major research projects emerged, each serving the evolving research area of DBEs: The DBE and the OPAALS. One aspect of both projects is the implementation of an Evolutionary Simulation Framework of DBEs, in order to provide a fundamental knowledge for DBEs and an interdisciplinary research platform for the community. The framework, named Evolutionary Environment Simulator (EvESim), acted, besides simulating biologically inspired P2P networks and the DBE itself, as a framework for understanding, visualising and presenting the DBE's concepts to partners within and outside the project.

The EvESim faced several challenges, namely i) overcoming the drawbacks of a complex model, ii) reducing the overheads coming from the reuse of the simulation toolkit REPAST, iii) increasing the simulation performance and iv) emulating the behaviour of a real P2P network, which were outlined in this and other papers (Kurz, T., Noguera, J., Eder, E. & Heistracher T., 2007). The main research question was if and how it is possible to overcome the identified challenges and issues and furthermore, which open-source P2P infrastructure can support a distributed EvESim as well as the DBE infrastructure. This interest of research led to a distributed and refactored version of the EvESim. To overcome the issue of a complex model of the EvESim and to provide a well-structured basis for the distribution, the REPAST toolkit, a rich agent-based simulation framework, was detached from the EvESim. Moreover, the EvESim was redesigned and structured in several different encapsulated entities. This led to a more generic model of the framework.

In order to achieve a distributed version of the EvESim, several core services were identified from the previously defined entities. These core services served as the main components of the simulation framework and were implemented on top of open-source P2P application servers, namely ServENT and Soapod, which provide a decentralised infrastructure. The P2P application servers, and the core services respectively, provide a valuable basis for distributing the EvESim in a P2P network. Consequently, the paper aimed at testing these two application servers of choice as well as documenting the implementation of rich services.

The ServENT provides a highly distributed approach to deploy services on a P2P network. It allows the user to implement services, deploy them and connect to the service by instantiating a proxy. Furthermore, the available ServENTs automatically connect to each other and provide the possibility to search for services within the whole network of ServENTs. However, the automatic neighbour lookup is restricted to a Local Area Network (LAN), but essential ServENT neighbours can be set up manually. Services themselves can be easily implemented and have access to basic information of the ServENT itself. Nevertheless, as the ServENT relies on proprietary protocols and infrastructures, it does not enable a high adaptability.

The Soapod provides an application server, which serves the DBE's concepts by extending commonly used Web service standards. As Soapod was implemented as an improved successor of the

ServENT, it overcame the issues of the ServENT in regards to adaptability and open standards and also provides service implementation, service deployment and service proxies. Although it provides a generic solution of implementing DBE services, which bases on highly adapted and common Web service standards, the creation of services is not as convenient as in the ServENT. In particular, the configuration of such services is very complex and it is not well documented at the developers' platforms. Several Soapods within a LAN are manually connected by entering the neighbours to the Soapod's configuration file. Nevertheless, Soapod provides better distribution capabilities in terms of service usage and service performance, at least as far as it is tested by now.

The ServENT as well as the Soapod provides a proper fundamental P2P infrastructure for the distributed version of the EvESim, and the DBE respectively. Although, the ServENT was implemented within the DBE project and was designed for the DBE, it faces several issues in regards to service performance and usage. Also Soapod did not provide the best solution for the DBE. Although, it serves the idea of global scale best, this led to a complex service implementation and in particular to a more complex service configuration than in the ServENT.

In conclusion this paper is not able to decide which open-source P2P application server, ServENT or Soapod, meets the requirements for the Evolutionary Simulation Framework best, but in regards to performance issues and high adaptability, it can be said that the Soapod presented a more stable infrastructure than the ServENT. Even though this work can not give a final answer, it provides valuable feedback to the open-source projects of choice. The feedback is given in terms of documentation, rich service implementation, bug fixes and suggestions for new features. It has to be proven for the future if the ServENT and the Soapod provide a stable P2P infrastructure for the simulation framework and for a DBE in general. Although, first tests showed a slight improvement in contrast to the stand-alone version of the EvESim, it still lacks persistent and stable simulation scenarios. Furthermore, the project consortium has to test and prove these two application servers of choice in regards to the DBE. This has to be done by using the Evolutionary Simulation Framework under real simulation conditions, such as critical mass and clustering simulation use cases (Kurz, T. & Heistracher T., 2007). It has to be shown if the shortcomings of the EvESim have been overcome by redesigning and distributing the simulation framework and if it provides a proper solution for the DBE's concepts.

In regards to the EvESim, acting as an interdisciplinary research platform for partners within and outside the project, it has to gain capabilities of visualising, importing and exporting data, and adapting simulation cases. Therefore, the Graphical User Interface (GUI) has to be extended toward these needs. Consequently, once more the EvESim has to provide the capability of loading social networks or behavioural settings from other partners as well as visualising the simulation network and its data. Furthermore, the model has to be implemented in a more generic way, which provides the possibility to adapt it to new needs of the research community, such as new simulation scenarios or new types of agents. In addition to that, improvements for the implementation of the underlying open-source P2P application servers, namely ServENT and Soapod, have to be integrated and evaluated in the simulation framework.

## Acknowledgements

This work was funded in part by the EU project OPAALS, contract number IST-034824.



This article is licensed under a Creative Commons Attribution-Share Alike 3.0 Unported License (<http://creativecommons.org/licenses/by-sa/3.0/>).

## References

- Adelberger, C. (2008). Evolutionary Framework Simulation (Diploma Thesis). Fachhochschule Salzburg GmbH, Salzburg, Austria.
- Alonso, G. et al. (2004). Web Services - Concepts, Architectures and Applications. Springer Verlag, Berlin Heidelberg.
- DBE Website. (2008). Retrieved May 23, 2008, from <http://www.digital-ecosystem.org>
- EvESim Website. (2008). Retrieved July 10, 2008, from <http://evesim.org>
- Gamma, E. et al. (1995). Design Patterns - Elements of Reusable Object-oriented Software. Addison-Wesley, New York.
- Goldberg, D.E. (1989). Genetic Algorithms in Search, Optimization & Machine Learning. Addison-Wesley, Boston.
- Google Maps API. (2008). Retrieved July 10, 2008, from [code.google.com/apis/maps/](http://code.google.com/apis/maps/)
- Kurz, T. & Heistracher T. (2007). Simulation of a Self-Optimising Digital Ecosystem. In: *IEEE First International Conference on Digital Ecosystems and Technologies*, 21-23 February 2007, Cairns, Australia.
- Kurz, T., Noguera, J., Eder, E. & Heistracher T. (2007). Peripheral Simulation Framework for Digital Ecosystems. In: *1st international OPAALS Conference on Digital Ecosystems*, 26-27 November 2007, Rome, Italy.
- Newcomer, E. (2002). Understanding Web Services: XML, WSDL, SOAP, and UDDI. Addison-Wesley, Boston.
- OPAALS Website. (2008). Retrieved May 21, 2008, from <http://opaals.org>
- Repast Website. (2008). Retrieved May 07, 2008, from <http://repast.sourceforge.net>
- Soapod Website. (2008). Retrieved May 14, 2008, from <http://www.soapod.org>
- Swallow Website. (2008). Retrieved May 14, 2008, from <http://swallow.sourceforge.net>