Digital Business Ecosystem

Contract N° 507953

Workpackage 9

Model of Fitness Landscape

Deliverable 9.2

Report on Evolutionary and Distributed Fitness Environment

Information Society and Media

Project funded by the European Community under the "Information Society Technology" Programme

Contract number: 507953
Project acronym: DBE
Title: Digital Business Ecosystem

Deliverable Nº: D9.2
Due date: June 30, 2006
Delivery date: 27th July 2006

Short description:
This report elaborates mainly on the set-up and simulation of a distributed Evolutionary Environment.

Author: Thomas Kurz, Giulio Marcon, Hisanaga Okada, Thomas J. Heistracher (STU) and Antonella Passani (Censis)
Partners contributed: STU, Censis, HWU, ICL, ISUFI, LSE, UBham, IBM
Made available to: Public

Versioning		
Version	**Date**	**Author, Organisation**
1.0	21/07/2006 (final submission)	STU
0.1	26/06/2006 (first submission)	STU

Quality check
1st internal reviewer: Gianni Dominici, Censis
2nd internal reviewer: Philippe De Wilde, HWU
3rd internal reviewer: Edgar A. Whitley, LSE

This work is licensed under the Creative Commons Attribution-NonCommercial-ShareAlike 2.5 License. To view a copy of this license, visit: http://creativecommons.org/licenses/by-nc-sa/2.5/ or send a letter to Creative Commons, 543 Howard Street, 5th Floor, San Francisco, California, 94105, USA.

creative commons

COMMONS DEED

Attribution-NonCommercial-ShareAlike 2.5

You are free:

- to copy, distribute, display, and perform the work
- to make derivative works

Under the following conditions:

Attribution. You must attribute the work in the manner specified by the author or licensor.

Noncommercial. You may not use this work for commercial purposes.

Share Alike. If you alter, transform, or build upon this work, you may distribute the resulting work only under a license identical to this one.

- For any reuse or distribution, you must make clear to others the license terms of this work.
- Any of these conditions can be waived if you get permission from the copyright holder.

Your fair use and other rights are in no way affected by the above.

Contents

List of Figures

List of Tables

Executive summary

This report describes a model of distributed optimisation and a simulator of the Evolutionary Environment (EvE) which is an extension of the Fitness Landscape simulator outlined in DBE deliverable D9.1. It also explains the current EvE service implementation in the context of the DBE architecture and the underlying concepts of distributed optimization and distributed intelligence that can be studied via the EvE simulator. A detailed requirements specification for the simulator as a result of the multi-partner collaboration is given ("bridging" between natural Science, COM, BUS and Social Science) as well as an introduction to the simulation framework used (RePast). Common interaction patterns, details of the SBVR encoding, and the description of the relevant implementation details of the EvE simulator precede a first discussion of preliminary simulation results addressing *critical mass*, *clustering*, and *clustering by usage* to better estimate metrics for the dynamic behavior of the DBE system.

1 Introduction

Das logische Bild der Tatsachen ist der Gedanke.
(Ludwig Wittgenstein, 1889–1951)

1.1 Aims of this paper – success criteria

This report should provide a more abstract view on the EvE although it emphasises on integration and collaboration issues regarding the EvE, in order to reduce the gap between basic research and application. The EvESimulator was intended to provide a framework where, for example, the basic research on genetic algorithms can be utilised and tested on networks which are as close as possible to reality. Additionally, the import and simulation of real-world network topologies and social interactions coming from social analysis is possible as well.

Moreover, D9.2 describes the development and model of the EvESimulator tool. It outlines only the connection to GAs but we did no in-depth research on GAs in regard to the EvESimulator. The outcome of our recent work on GAs can be found in [1]. In the EvESimulator we only use a very rudimentary implementation of a GA which can easily be replaced by a more sophisticated one in future. The representation of SBVR is also only outlined because the abstraction of SBVR as services with attributes is the closest model we can think of in order to provide also a model for other partners they can work with.

The EvESimulator and this report, respectively, shows the general feasibility of the distributed optimization approach in the DBE. It seeks to improve general understanding of the complex interdisciplinary interactions in DBE and explains the benefits of having the EvE as architectural component to different stakeholders. Beside that, the EvESimulator helps with the configuration of typical business environments and networks to explore fitness options for SMEs and moreover, provides a tool that can be used publicly outside the consortium to convince 'critical' people of the benefits of DBE utilisation.

1.2 Document structure

Chapter two, the EvE service implementation should summarize the basic concepts of the EvE and EvE service implementation (D9.3, D9.4) in order to set the context and making the deliverable readable without reading through all the EvE documents. References are given for further information on the EvE high-level architecture as well as the EvE service implementation. Additionally, the main concepts like migration for example are outlined and set into the context of this report.

In Chapter three we start with a short recap on the objectives from the technical annex to show where to find the information stated in the technical annex. Initially the work of the EvESimulator was planned to be much more rudimentary. At the review in January 2006 the visualisation capabilities and possibilities in terms of merging results of different SCI partners and utilising them in a joint framework were highly appreciated by the reviewers and so much of the work done for the EvESimulator was integration and collaboration. Consequently, Chapter three sums up the collaboration efforts of the different partners and presents the background of the joint model.

Chapter four deals manly with the model and implementation of the EvESimulator. Furthermore, it provides a short introduction and a kind of handbook for the usage of the graphical user interface. The EvESimulator was shown in January to representatives of the European Comission and we changed the model afterwards to enable a broader collaboration. In order to show the change in the initial model and the reasons for that, we outlined the January Version in the first section of Chapter four. The second section presenting the model of the EvESimulator, discusses the current status of the EvESimulator model and the main parts of this model. As the EvESimulator is a framework which enhances a reuse of functionalities developed by different stakeholders (genetic algorithms, social networks analysis, global optimisation, etc.) the model is in some kind the intersection of the simulation models done by the science community of the DBE. The last section of this chapter is the user interface documentation of the EvESimulator including the outline of the implemented features.

Before the conclusion sums up the work done in regard to D9.2, Chapter five deals with the first results of the simulations. Because of the limited resources it was not possible to do an in-depth study of several simulation cases of interest. Nevertheless, three of the most frequent questions regarding critical mass and clustering were already investigated using the simulator and Chapter five presents these first results.

The reason for implementing these simulation cases is motivated by the fact that we wanted to show the capabilities of the EvESimulator and get indications for further research in this field. The utilisation of the EvESimulator by other partners will be described in their deliverables, respectively.

1.3 Audience and usage of this document

The manifold collaborations and integration work done in regard to this deliverable made it necessary to structure the deliverable not only in a report about one line of research, but in a document that can be read and understood by reading it from the first to the last page and also particular chapters can be read separately.

Chapter two is a summary of the EvE service implementation. The target audience should at least have a basic understanding of the technologies utilised in the DBE project. Additional references are given in order to read more about certain technologies and concepts used in the EvE.

Chapter three is a non technical chapter that can be read independently from the others. Experts of all disciplines can read this chapter without detailed technical knowledge. It shows the the relations to the technical annex to agent-based modeling in general as well as all the current collaborations with other disciplines.

Chapter four and five describe the EvESimulator and the results of the first simulation cases. Therefore, at least some basic knowledge of the DBE and the EvE architecture is necessary. Nevertheless, the target audience are also researchers from social science or non-technical experienced people with an interest in simulation of SME networks. Because of the rich configuration capabilities of the EvESimulator it can be used also without programming knowledge and various simulations can be processed without writing one single line of code.

2 EvE Service Implementation

The following chapter intends to be a brief summary of the EvEService implementation. Details can be found also in [2], [3] as well as on the respective sourceforge project [4]. Nevertheless, a short introduction into the core concepts of the DBE are sketched in order to have a consistent view on the most important implementation issues.

Instead of having one centralised management system handling all the systems in its network, the DBE has been designed to mimic the evolution of interrelated and dynamic subcomponents found in nature. Beside the theme of an ecosystem, the DBE technology being developed uses concepts from the fields of natural science, social science, computing science, business, as well as economic development.

Services in the DBE are described under two headings: Business, and Computing.

Business models seek to describe the service in business terms i.e. the company's goals, products, price models, business model and strategy. All of this information is defined in terms and concepts that are abstracted from any implementation specification. The goal here is to enable non-IT specialists to define and understand such specifications and at the same time make them precise and complete enough to be computable. The computable form of the business models provides the ability to make contract negotiations and agreement and compute the merit of business offers.

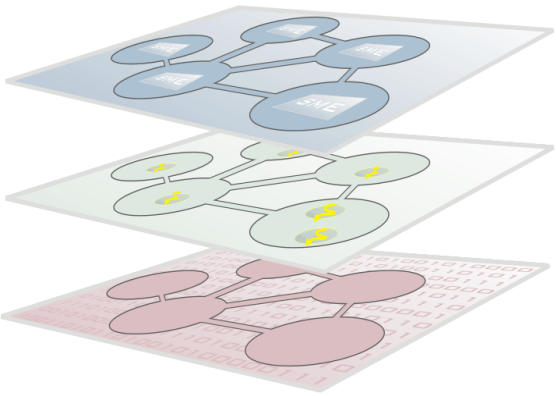
The *Computing* models associated with the various business services enable business offers to be consumed via online services. The model enables non- predefined IT functional modes to be integrated into the DBE. The goal is to be as un-invasive as possible and avoid enforcing compliance to DBE functional and technical models.

These business and computing models together with "instance" data such as location, name and cost makes up the "Service Manifest" - a self-contained descriptor of the service.

2.1 Architectural Overview

The DBE bases on an open source environment with a set of tools for modeling, communicating and data processing. One of its aims is to allow all SMEs to interact with each other which was not previously possible; even for those without experience in ICT. The architecture of the DBE is made up of three distinct layers. The main components in this three layer architecture are:

1. The Service Factory
2. The Execution Environment
3. The Evolutionary Environment



The *service factory* (SF) is essentially the union of all the DBE Studio programs used in the system, including the distributed systems like the Model Repository, Testing Environment, Composer and the Model Recommender. As is the case for most software in he DBE, the software development is done in Java (see also [5]).

The *execution environment* (ExE) is the union of all the operative systems found in the network. It is the engine that enables services to be introduced, managed and consumed within the network. Also, the structural features such as authentication, authorisation, privacy, transactions, and account management are implemented in the ExE layer of the DBE architecture (see also [6]).

The *evolutionary environment* (EvE) is the union of all the Habitats and is responsible for the long term management of the services found in the network. Based on the successes and failures in the migrations and the usage histories found within the network, the EvE is the dynamic system that adapts to the constant changes found within the network. It works to address intelligently recurring requests in particular regions in the EU to optimise a given region or Habitat. For more information on distributed intelligence system (DIS), refer to the DIS section later in this chapter.

2.1.1 Service Oriented Architecture

The network itself is a *service oriented architecture* (SOA) for each Habitat. This gives the chance that a large number of programmers can create, combine or improve distributed applications through the use of standardised interfaces. In the case of the DBE, this will allow the system to automatically evolve over time to better suit the evolution of the DBE network.

The EvE network is essentially the union of all the Habitats based on the SOA. This is implemented at the Habitat level where instead of having programs working at the network level. The Habitats connect to each other to migrate services using a P2P structure based on a servent infrastructure.

The EvE network is the union of all Habitat nodes within the P2P Network. A Habitat node is a DBE software service that runs on top of a Servent. Therefore, the EvE does not implement its own networking algorithms, but uses the underlying P2P system infrastructure.

Each Habitat must actively perform these tasks:

- Discovery of new Habitat nodes
- Connect to a Habitat node using the node's unique ID
- Provide a proxy for a connected Habitat

These three features are already implemented in the Evolutionary Environment Network implementation [4].

2.1.2 Habitat Architecture and Implementation

Services themselves are implemented into the system as a description and an executable reference to the actual service. These service descriptions showcase which features the services are able to perform and are formatted in a way to be computable for a given user request. As they are small and portable, services can easily undergo migration by simply copying the contents of a service from one service pool to another using an existing connection. Based on the success of a service it can migrate to other Habitats in order to find another niches in the ecosystem i.e. other SME where it proves useful.

Evolutionary algorithms are key in the DBE in two levels. One way for utilising evolutionary algorithms would be that individual requests for features from a user

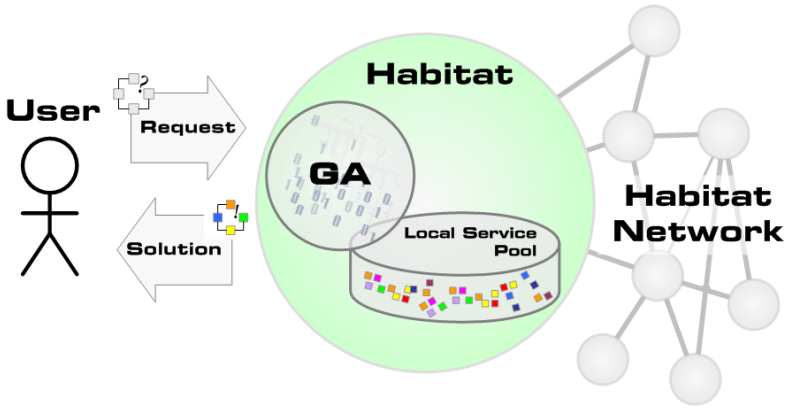


Figure 2.1: Representation of a SME and Habitat, respectively within the Evolutionary Environment (EvE)

are to be fulfilled with a list of viable services for the request. This is done using genetic algorithms to find the appropriate services in its own local service pool, within its Habitat. Another way that evolutionary algorithms are to be used is for the network of Habitats. Services are introduced into a Habitat and they have the ability to be copied or migrated to other Habitats within the network. This is done partly through the evolutionary environment and enhanced by distributed intelligence system which is currently being developed.

2.2 Concepts Used

This section summarizes the most important concepts of the EvE. Parts are already implemented and some are only simulated in the EvESimulator framework. Moreover, related tools are explained in brief.

2.2.1 EvE Habitat Service

Also: Habitat, Habitat Service or Node

To a single user, a *Habitat* is a means to store and manage and access a family of services. To the Evolutionary Environment, Habitats are dynamic, interconnected nodes that communicate with each other. At present, a one-to-one relationship exists between an SME and a Habitat.

Figure 2.2 visualises the main modules of the EvE design. These are namely,

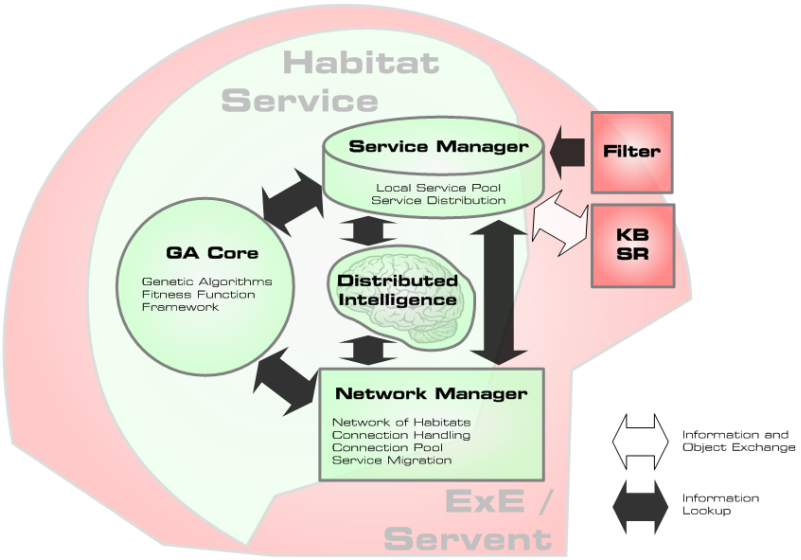


Figure 2.2: Schematic visualisation of the EvE Service implementation architecture.

the network manager, the service manager and the intelligence modules (Genetic Algorithm (GA) Core, Distributed Intelligence System).

The *networking module* is responsible for building and servicing the Habitat network. A Habitat node is an independent DBE software service that runs on top of a Servent, P2P network acting as a server and a client for available services with other Habitats. It is independent because the Evolutionary Environment (EvE), the network of interconnected Habitats, does not implement its own networking algorithms, but uses the underlying P2P system infrastructure of the Habitat.

The *service module* manages the tasks for its own local service pool, the collection of services that can be offered within the Habitat.

The *intelligence modules* are made up of two evolution based components: the genetic algorithm (GA) and Fitness Function Framework (FFF). These modules processes a user request for services. Given a service request, the GA and FFF work together to try to find the appropriate combination of services found in its current local service pool. For more information for the GA and FFF, please refer to deliverables 16.3 and 9.1. The EvE will be enhanced in future by introducing the distributed intelligence system (DIS). The DIS works with the migration component of services found in the Habitat. It works to migrate services throughout the network so that it can find better suited Habitats for it to go to. For more detailed information on the DIS see also [7].

For a user their habitat is the gateway to what is wanted, in other words, the collection of services specific for his needs. A Habitat houses its own Local Service Pool, the collection of services that can be offered within the Habitat, as well as all the software that manages it.

2.2.2 EvE Service

There are two types of service models in the DBE: business and computing.

The aim of the *business* service is to be able to describe business services so that the concepts are defined and understood by non-IT specialists as well as being unambiguous and complete enough such that they can be computed. The services can be anything from the company's goals to products, price models, business models and strategy. The computation of the business service will be able to produce contract negotiations, agreements as well as computing the merit of business offers.

The *computing* service works with the business service so that they can be managed by online services. This allows previously undefined IT functional modes to be integrated into the DBE. It is desired that the computing service is not intrusive and to steer clear of the functional and technical models found in the DBE.

The business and computing models, along with such data as location, name and cost, are the components of the Service Manifest. A Service Manifest is a unique descriptor of the service.

2.2.3 Migration

In nature, it is essential for some organisms to move to other physical locations to be able to find their own specific niche in the ecosystem (succession). In the DBE, services must also be able to move from one Habitat to another. This is known as *migration* and is the concept used in the DBE to allow services to find their niche Habitats in the EvE. A service is introduced in the DBE when its service manifest, the description of the service, enters a Habitat's local service pool. The information is distributed to other Habitats found in the network and this process is called *service migration*.

The migration of services is determined primarily by the *migration probability* between the habitats and its *service usage history*.

Migration can be random or targeted. Each Habitat connection has an associated migration probability. Migration occurs by sending out services from its service pool

based on service usage. For more detailed information see also [7] and [2].

2.2.4 Service Pool

A *service pool* contains all the services for a single Habitat. Although connected, each Habitat is independent from each other in that a service pool for one Habitat cannot be accessed directly from another Habitat when fulfilling requests. A new service can be introduced to a service pool or through migration from other Habitats. Used services tend to proliferate through usage in comparison to others.

A Habitat's service module is what is responsible for managing the services within its service pool. A service pool is stored on disk at regular intervals and is read each time the habitat service is to be restarted.

It is at the Habitat level that it is decided whether to migrate successful services or to delete less successful services.

2.2.5 Local Optimisation

Semantics of Business Vocabulary and Business Rules (SBVR) is a structured version of natural language which is unambiguously understood by the computer as a means to explain a business model. The obvious benefit of having it in a natural language would be that highly specialised documents and diagrams can be broken down into SBVR and be understood by non-IT members of the project. As such, SBVR is being integrated into the DBE project by using the natural language for software specification at the development stage, and can also be used to provide the users of the DBE with the interface needed to attain the services requested.

In the context of a single Habitat, service descriptions of the services are to be represented in SBVR where not only a human user can read and understand the service, the system can decompose the description unambiguously. A user in the system can then query the Habitat using the same SBVR language which will then allow the genetic algorithms within the Habitat to try to produce a solution set.

2.2.6 SBVR Editor

In the context of the DBE, the SBVR editor is to be used by the individual users of the system who would like to request services for their needs. When the request is made in SBVR, the EvE can then employ the optimisation algorithm to process and output the services to fulfil the request.

The editor is also to be used by developers of the DBE in the future so that business models are represented in SBVR. The advantage of this is again so that non-IT members of the project can understand these models and to work on the concept of code generation via SBVR. This is done just as UML can be used to generate code.

3 EvESimulator Basic Conditions

The intention of the EvESimulator, from the beginning on, was not only to simulate the behaviour of the Evolutionary Environment, but also to provide partners from natural science, social science, business and computing a framework to collaborate and test their findings together. Based on the objectives stated in the task description of *C37 Simulator Evolutionary Environment* the following chapter elaborates on the basic framework used for the EvESimulator as well as describing briefly the collaborations with the partners, directly involved in the EvESimulator modeling.

3.1 EvESimulator Requirements and Expectations

Referencing the DBEs Technical Annex [8], the EvESimulator should act as a bridge from science to computing and should help to provide findings for the implementation and configuration of the Evolutionary Environment on top of the DBE Servent infrastructure. Table 3.1 gives a brief overview of the initial objectives of *C37 Simulator Evolutionary Environment* and indicates where the most important information can be found.

Although the initially defined objectives were intended to implement a simulation of the EvE, the current implementation offers not only a simulation but also a rich framework which can utilise various outputs of science and acts as an platform for further research of digital business ecosystems. Furthermore, it can be used also as a presentation tool for new regions in the future. With the migration to [9] in July 2006 and the further research in *Open Philosophies of Autopoetic and Associative Digital Ecosystems - OPAALS* basic conditions for the sustainability of the framework are provided as well.

Realisation of a SME-representing container for services and service chains	Section 4.1 Section 4.2.1 Section 4.2.2
Implementation of fitness calculus and simulated fitness update in the distributed evolutionary environment simulation	Section 4.1 Section 4.2.1 Section 4.3.2 Javadoc org.dbe.eve.ga D16.3 [1]
Integration of concepts and mechanisms from S12	Whole document
Configuration of typical Business Environments (simulated data) to explore fitness options for SMEs	Section 4.2.4 Section 4.2.5 Section 5.1 Section 5.2 Section 5.3
Imitation of the evolution in the simulated DBE (e.g. boot-strapping) and comparison of concepts and methods (e.g. long-term stability)	Section 5.1 Section 5.2 Section 5.3

Table 3.1: Outline of the objectives planned and documented in the technical annex.

3.2 Recursive Porus Agent Simulation Toolkit - Repast

Although we could have implemented the agent based simulation needed for the EvESimulator from scratch, we decided to build up from a well known and already widely tested framework. We decided to take a practical approach for framework evaluation. Beside the literature study of agent-based economy simulations (see [10, 11, 12]) we took part at the International Workshop on *Agent-Based Models for Economy Policy Design - ACEPOL05* where the world leading experts of agent-based economy simulation met. Consequently, we decided to choose the Recursive Porus Agent Simulation Toolkit - Repast (see also [13, 14]).

Repast is a software framework for creating agent-based simulations using the programming language Java. The framework consists of a library of objects for creating, running, displaying, and collecting data from an agent based simulation. A Repast simulation is primarily a collection of agents of any type and a model

that controls and sets up the execution of these agents behaviours according to a schedule. This schedule controls the execution of agent behaviours as well as actions within the model itself. These actions can be: updating the display, recording data, and so forth. The Repast model is responsible for setting up and controlling simulation visualisation, data recording and analysis. Repast was initially created by the Social Science Research Computing at the University of Chicago as a library of Java classes that should interact, work together with, and simplify the Swarm simulation framework.

The typical Repast model contains a set of agents. These agents can on the one hand be homogenous or heterogeneous, and on the other hand the agents are arranged in a hierarchy. Regardless of their composition, each agent has behaviour, the interactions of which a modeller is interested in exploring. The agent behaviour is executed at every simulation step (tick), and what gets scheduled then is an event or, as it is called in Repast, an action.

Java allows the programmer to organise his or her code into packages. The Repast framework consists of 210 classes organized into 9 packages as well as several demonstration simulations. Repast includes several varieties of charts for visualising data such as histograms and sequence graphs. Furthermore, Repast can even take snapshots of running simulations and create QuickTime movies of simulations. The Java Enterprise Simulator borrows the visualisation from the Swarm framework as well as much of the design of the Swarm simulation toolkit and can properly be termed a "Swarm-like" simulation framework.

The intention of the consortium of implementing the infrastructure and tools in Java led to another promising framework based on Swarm, the Java Enterprise Simulator Program - jES [12]. It describes in a detailed way a two sided world, considering both the actions to be done, in terms of orders to be accomplished (the What to Do side, WD), and the structures able to do them, in terms of production units (the which is Doing What side, DW). The simulation model is, first of all, a description of the enterprise, as it is. jES executes exactly what the users suggests what has to take place into the simulated enterprise in the model. Although the approach sounds interesting, the provided features and good support of Repast though the sourceforge mailing list lead us deciding for Repast. Nevertheless, an in depth assessment of jES' features proves useful for future research. For more information on jES see also the paper by [12].

Besides the visual capabilities of the toolkits, the scheduling mechanisms were

one of the most important decision making criteria. The agent based simulation framework Repast behaves as a discrete event simulator whose slots of time are known as ticks. A tick exists only as a hook on which the execution of events can be hung, ordering the execution of the events relative to each other. If no events are scheduled for a certain tick, then it is as if that tick never occurred. Scheduling in Repast works statically and dynamically. That means that scheduling can be manually implemented by the modeler (static) or automated via the model (dynamic). The Repast scheduling mechanism allows more sophisticated dynamic scheduling, for example the execution of an event can itself schedule other events executing in the future. The scheduling mechanism in Repast is responsible for all the user-defined state changes in the simulation. Scheduling consists of setting up and executing actions at some specific time, with the help of the so called ticks, relative to other actions. As implemented in Java, this means setting up and executing method calls on objects at the specific time [15].

3.3 Collaborations

Although, many other partners contributed to the EvESimulator in form of comments, suggestions and modeling, these are the most related ones during the last months. Additionally, some partners signalised their interest in future cooperation.

3.3.1 Optimisation Algorithms (UBham)

The first version of a genetic algorithm utilised within the EvESimulator was developed with Dominique Chu (UBham) during a workshop in 2005. During the annual review in January it was agreed to develop an updated version with John Woodward (UBham) by March and include it into the EvESimulator framework. UBham would investigate properties of this algorithm as part of WP8 Dynamics, Selection and Aggregation. It was intended to be easily modifiable to the needs of the EvESimulator so that we can use it straight forward without much additional effort. UBham studied the performance of several GAs on various tests, and recommended the UMDA algorithm of Heinz Muhlenbein (see also [16] and [17]). Unfortunately, the researchers working for UBham on this task left the project in April, and UBham formally left the project in May. Consequently, the current implementation in the EvESimulator is a modified version of the first draft version developed in the mentioned workshop in 2005. The modifications of the algorithm

were done in addition to the implementation of the EvESimulator and initially not a task of STU. Therefore the implementation is, though working without problems, still in the draft version. Because of the modularity of the EvESimulator any other optimisation algorithm can be utilised in the future in order to enhance the efficiency of the simulation framework. Further information concerning our internal research on optimisation can also be found in the paper by [1].

3.3.2 Global Optimisation (ICL)

ICL did extensive research on the set up of an digital ecosystem. The presented work within *WP12 Optimisation* resulted beside others in a snapshot of a simulated digital ecosystem in textual form. Beside the SMEs and their services, relationships between SMEs in form of trust were modeled. In order to utilise the output of this extensive research the EvESimulator was extended in order to model trust and take also input files for further processing. Two point have to be emphasized here in order to understand the level of interaction and collaboration.

First, the algorithms of building up the simulated DBE of ICL were implemented in the programming language C, whereas the EvESimulator is implemented in Java. As a consequence we had to identify the interfaces for a collaboration without spending too much time in changing the existent code base. The solution was to leave the C code base for setting up the network as it was and take the textual output file as the linking part. Additionally, that offered ICL the possibility to do their further research on parts of WP12 by using the code that already exists and take advantage of the EvESimulator where it makes sense.

Nevertheless, as the second point which should be emphasised here, the output file has to be modified by ICL to meet the specifications of the EvESimulator. The EvESimulator is based on the idea of using state-of-the-art technology as well as best practices of software engineering and consequently provides a generic framework. Therefore, the use of XML was one of the key technologies used for data input and export. For this reason examples of the XML files as well as XSD schemata were exchanged with ICL. The adaptation of the output files resulted in additional work for the group of ICL but opens the possibility of taking full advantage of a dynamic simulation of DBE networks within the EvESimulator without modifying most of the existing code base.

For a one-to-one mapping of the data set provided by ICL, further modifications would be necessary. The drawback in this case was that the representations of

numeric attributes as real numbers ($\mathbb{R}$) was flattened to a representation in natural numbers ($\mathbb{N}$). If a model using $\mathbb{R}$ proves necessary for the dynamic simulations, the framework can be extended by this feature. For the simulation of SBVR it is not needed because SBVR represents models which can be flattened into a set of $\mathbb{N}$ attributes.

3.3.3 Symbiosis and Competition (HWU)

As the implementation of HWUs simulations were Java based and the model fit well into the planned EvESimulator framework the collaboration with HWU has been planned not only on an data exchange level but on an utilisation of existing code itself. Because of the considerable changes in the EvESimulator from January on, the use of the simulator code base will be possible only for the last part of the project. The planned activities concerning this cooperation is at least the run of one of HWUs simulations within the EvESimulator framework.

An introduction of a list of demanded services substituting the formerly implemented SME profile was necessary. The already implemented portfolio of services (offered services) had not to be changed. Moreover, the EvESimulator model was set up to easily support the needed functionalities of HWUs simulation cases. Additionally, following further extensions of HWUs simulations should be possible or even accelerated when using parts of the EvESimulator:

1. Introduction of money into the simulation
2. Bootstrapping of Ecologies (joining / disappearing of SMEs)
3. Introduction of trust
4. Study of effects of local versus global optimisation
5. Introduction of service chains instead of services

Regarding these planned activities, it is possible to introduce money in principle; the implementation itself is up to the partners, respectively. Activities 2, 3, 4 and 5 are already in part or fully implemented into the EvESimulator and can be accessed at the moment from STU's SVN server and from July via `evesim.sourceforge.net`.

3.3.4 Business Group

The necessity for a close collaboration and information flow with the business group of the DBE project was apparent from the beginning of the EvESimulator specification on. Therefore, STU took part at several business meetings and collaborated especially with the leader of the business group Elmar Husmann (IBM). The wish for closing the gap between fundamental research of abstract problems and real-world scenarios become a strong understanding of the SMEs networks. The main findings of this collaboration can be found in Section 4.2.4.

3.3.5 Social Networks (Censis)

The collaboration between Censis and STU has been focused on the possibility to transfer knowledge developed by Censis on engaged SMEs to the EVESimulator. Censis research in Wp27, in fact, focuses on SMEs relational networks and - thanks to network analysis - visualises the correlation among SMEs and other local players.

A first result of Censis-STU collaboration was the transfer of data done in December 2005 for the DBEs second annual review. Those data were related to RC and Driver SMEs only (see [18]) and didn't impact the general structure of the simulator (variables, SMEs profile, algorithm, etc..), now that implementers and users have been engaged, new data will be available and will be integrated in the simulator thanks to the input/export features provided by STU. A first impact on the Simulator structure is now visible. The connections typologies Censis studied so far have been already introduced in the simulator and are reported in section 4.2.5. These connections typologies go from personal contact and sharing of resources as maximum of connection among SMES to more sporadic or absent relations. As already mentioned, those networks are not networks of services (pieces of software migrating from one ambient to another) but relational and business networks of SMEs engaged in the DBE. Nevertheless, the two layers - network of services and real word connections - show important points of contact. For example, three SMEs involved in several projects together may wish to share an agenda and look in the DBE platform for an agenda synchroniser. Besides this, face to face or business collaborations can have an impact on the level of trust between two enterprises. A high degree of trust, consequently, may invite one SME to prefer services provided by an already known enterprise instead of an unknown entity by this way introducing an important element in the migration pattern. In both examples a real word

connection impacts a digital activity and service exchange.

Besides this, the collaboration developed so far provided a common ground for understanding and developing a common language. This is a first step for real interdisciplinary research.

Censis future research will then provide more real word data. First of all, more data on Drivers' networks and then on implementers and user networks. Further research will have a positive impact on SMEs profile development too by introducing real word scenarios. Censis will try to visualise, if it exists, a possible definition of different actors (drivers, implementers, users, other local actors) in terms of interactions, i.e. try to understand if there exists a connection between the actor's role in the network and its level of interaction/collaboration with other local players. Besides this, Censis will provide a sort of typology of SME profile in term of business domain, business organisation and service possibly to be requested (see section 4.1). It'll be important then, again thanks to the collaboration with STU, to understand the possible relationship between SMEs profile and service migration rate. This will require further analysis but will be of great impact for the simulations itself. Then it is interesting to consider different advantages that different DBE partners can take of the simulator. From a computing perspective this is an important tool for visualising positive aspects of EVE and SBVR. From a social science and a training and RCs perspective, the simulator can become (if further improved from a usability point of view) an interesting instrument for explaining to SMEs and regional players the relevance of collaboration and of DBE. By modulating SMEs' profile and other contextual variables, it's possible to show which are the positive mechanisms of knowledge sharing, collaboration and clustering. This possibility is not yet in place but represent a future scenario for Censis-STU collaborations.

4 EvESimulator Implementation

After drafting the first model of the EvESimulation which was presented at the annual review in January 2006 in Tampere, Finnland, the following implementation chapter is devided in two main parts. First, the overall model is described in detail. Most parts of the model are already implemented, whereas some others are indicated as future extensions of the model, especially for a more lifelike representation of actors. Second, the features of the EvESimulator are documented following the structure of the graphical user interface. Additionally, the concepts and algorithms related to the configurations are explained.

4.1 Abstract Encoding of SBVR as RNA Strings

The following subsection describes the first version of the EvESimulator. Although the intention was to keep this representation for the simulation, we discovered that a redesign was necessary because especially science partners wanted to contribute to the simulator more actively after the first presentation of the EvESimulator at the 2nd annual review of the DBE project. Furthermore, we got a better understanding of SBVR[1] and the possibilities of flattening and abstraction, respectively. The need for a common representation model of SMEs and services led to the current EvESimulator implementation which is more a simulation framework for various applications than a single simulation of one use-case. Nevertheless, the first simulation model should be documented for history reasons as well as to show the need for a deeper investigation of modeling together with several partners.

Given that in the long run the EvE was intended to act on SME and service descriptions in SBVR we had to find a way to abstract this approach in an appropriate way. Keeping in mind that a considerable number of experiments on genetic

[1] `http://www.omg.org/docs/bei/05-03-01.pdf`. See also Stan Hendryx, "SBVR and MDA", *Business Rules Journal*, Vol. 6, No. 10 (Oct. 2005).

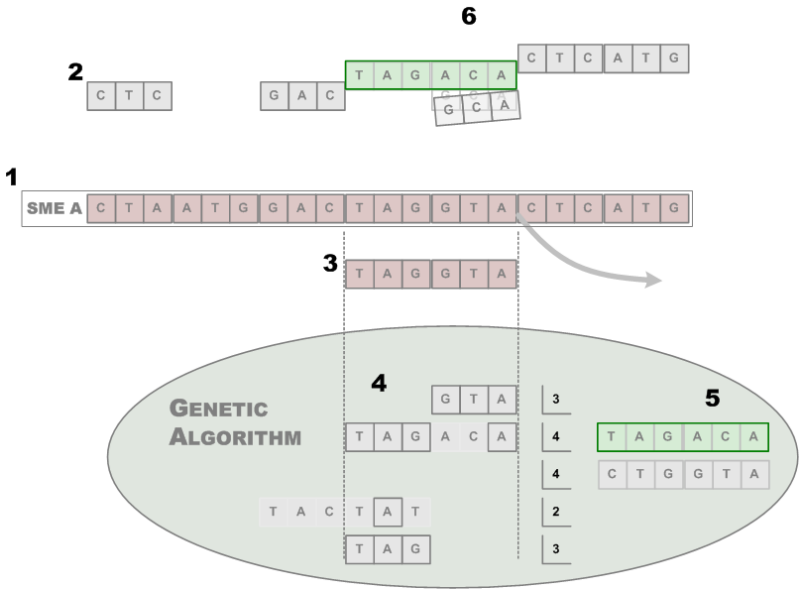


Figure 4.1: Basic encoding of the EvE Simulator

algorithms are done on a bit string representation of services and that we had to integrate this work with the simulation and the EvE, a compromise had to be found. As we are using similes from biology as well as we see the future SBVR descriptions like a DNA for services, we encoded the SBVR descriptions as RNA strings. These RNA strings are made up by so called codons which are tri-nucleotide units encoding a single amino acid in nature. In our case the codon is the smallest entity of an SBVR description that a service or SME needs to express its profile. Beside the simile regarding biology and the expected closeness to SBVR, an RNA string representation is very easy convertible into bit strings and vice versa, which would had made a later integration of bit string based genetic algorithms easy.

Figure 4.1 shows the principle encoding and optimisation wich was shown at the 2nd annual review of the DBE project. The following steps and actors are involved in a local optimization done at each of the SME agents in the network simulation. This process focusses only on the evaluation and representation done by one SME agent.

(1) The profile of an SME could be modeled as a RNA string. It represented a kind of identity of the SME in terms of business domain, processes as well as services possibly to be requested. As the SME profiles as well as the services will be described within the EvE using the same language, namely SBVR, the services

in the EvE were also represented as RNA strings. In future expansion stages of the simulation different types of SMEs were invisioned that could be encoded through the usage of certain characteristics in the RNA as well as in the interaction behavior. With regard to the social networks we needed here a clear picture of the interaction patterns of the actors involved, e.g. Drivers, Implementers and Users of the DBE because their portfolio and interaction behavior differs significantly.

(2) Parts of the profile information encoded in its RNA were supported by services used within the company, e.g. processes are enhanced using IT services. The productivity of the SME was evaluated as a function of coverage of the SMEs RNA by supporting services. This function had to deal, for example, with redundancy issues also and the productivity as a function of coverage of the profile by enhancing services from the DBE is not sufficient but for the first implementation we simplified the problem space to this comparison.

(3) Requests for new services by the SME were simulated by cutting a piece out of the RNA strand and requesting services or service combinations according to the encoded information. One example could have been the request for a accounting software which is requested by a SME to enhance its internal process.

(4) By applying genetic algorithms (GA) best suited service combinations were searched for in a pool of available services. The SME profile, the request as well as the service descriptions were formulated as RNA snippets.

(5) The optimization was coming up with the best candidate solution of service combinations - highest fitness count.

(6) The formerly used services at the requested point were released and the new service combination was put into place. Existing services can only be replaced if their functionalities are entirely covered by the new service. If not fully covered a overlap of functionality remained which was calculated as a cost function into the SMEs fitness calculus.

The process and representation of services and SMEs outlined in this subsection was the first version presented at the 2nd annual review of the DBE project in January 2006. Because of the already emphasized reasons we decided to redesign the model and implement the agents as documented in the following sections.

4.2 EvESimulator Model

In order to enable partners from natural science as well as business and social science researchers to utilise the EvESimulator, a common model for the EvESimulator had to be found. In the following section the model for representing services, SMEs and relationships between SMEs are described in detail. Although, the EvESimulator model is intended to be as close to reality as possible, the model represents an abstraction layer, which enables small real-world networks as well as well-defined problems in high scale networks. Moreover, the following model is a result of a number of collaborations sketched in Section 3.3 and is therefore considered as common sense at least within the related partners, though models may change for future applications.

4.2.1 SBVR Encoding and Service Representation

The new representation of SMEs and especially of service descriptions within the EvESimulator are closer to the actual SBVR descriptions as the RNA string encoding. It is a mapping of SBVR logic into a set of features (flattening), which results in a simplified model that does not take into account the full set of SBVR capabilities. Nevertheless, this model is a compromise between the real SBVR representation and the abstraction level that facilitates a simulation that is close to reality. Additionally, the matching of SBVR models and its theoretical implications are still being researched in the current stage. Consequently, a level of abstraction has to be defined so that a generic objective function can be defined, capable of being applied to a broader set of service descriptions (that is, potentially any version of BML).

As delineated as "A" in Figure 4.2, each Service is represented by a number of attributes. These attributes can be symbolic (colour of a car) or numeric (discount for a price). As symbolic attributes can be simplified by using $\mathbb{N}$, the range of attributes could be chosen as $\mathbb{R}$ for both, symbolic and numeric values. As SBVR describes models and the search will be also on the base of models, numeric attributes are not the main focus of future SBVR descriptions. Therefore, the values of attributes within the EvESimulator are currently set as a subset of $\mathbb{N}$, which means *integer*.

In case of service combinations the attributes of the individual services are merged and consequently build up a new service description. For better introspection, service combinations could be done in form of a tree structure in future, so that the information of the original atomic services persists within the new service.

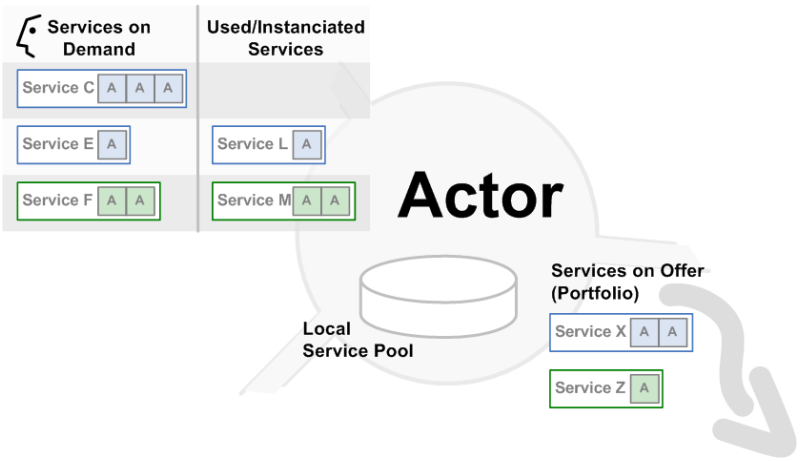


Figure 4.2: Representation of a SME and Habitat, respectively within the EVES-imulator called Actor

At the moment information of the original providers is lost when a new service is created and instantiated at a SMEs habitat. The comparison of two services is in any case a comparison of all the attributes of one service to all the attributes of another service and service combination, respectively.

When a new service is produced it appears first in the portfolio of its home-habitat. The home-habitat is managed by the actor which produces or offers a service. In case of a service combination of two existing services the actor who combined the service becomes the owner of the new service, because we assume that additional effort was needed to combine two existing services. Although the home-habitat for the new combination is assigned to the assembler, the service is not added to the portfolio of the actor. Information about the home-habitat as well as the migration history is also part of the service description.

4.2.2 Types of Actor

An actor in the EvESimulator represents one SME within the social network of a certain region. Since the mapping of SME to habitat has a cardinality of 1:1 at the moment, we are using the terms SME, habitat and actor interchangeably. Considering the existing social networks and the different actors within the DBE, five types of actors can be defined (Figure 4.3). They can differ in 1) the types of services they offer, 2) network growth for a certain type of actor over time, 3) the role of the local service pool and 4) the role of the user profile (e.g. services on

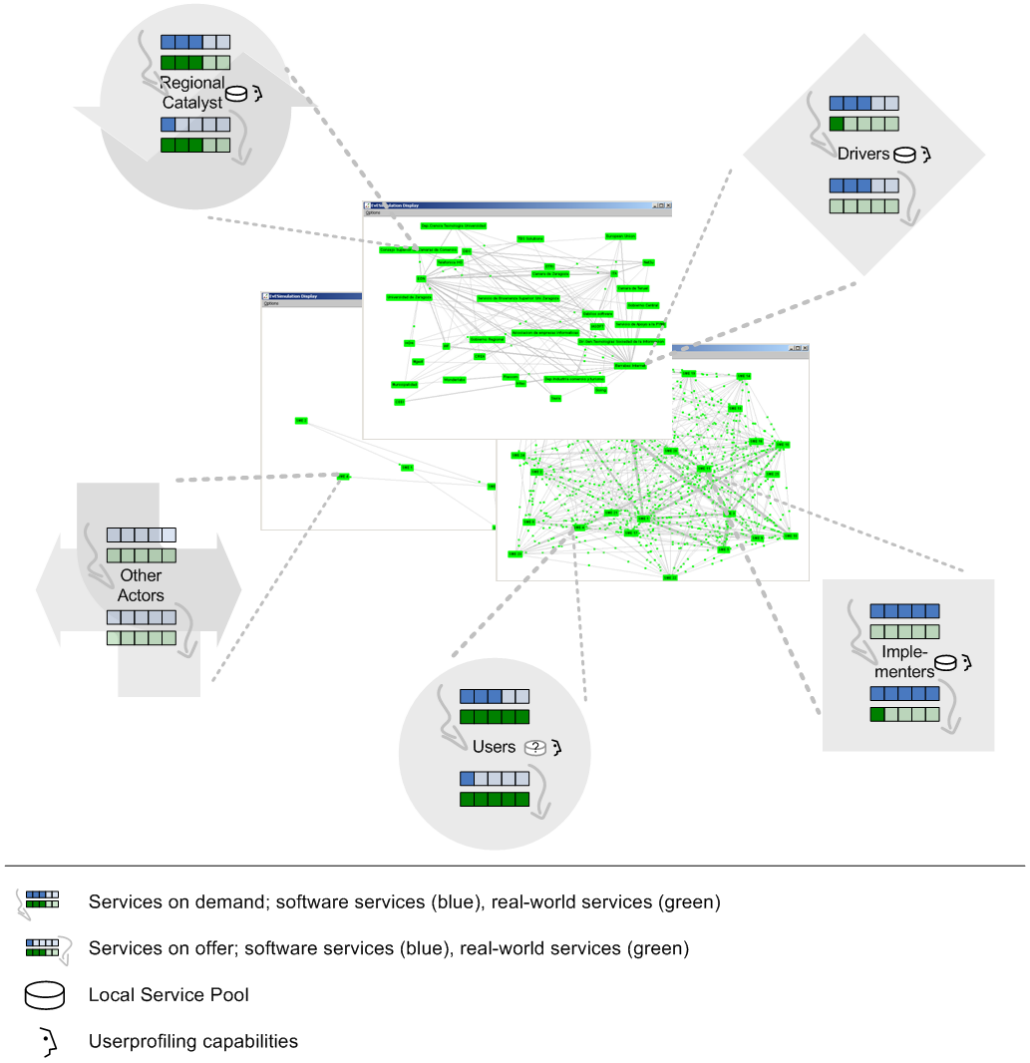


Figure 4.3: Different types of actors in the DBE network.

demand and offer).

As already shown in Figure 4.2, each actor has a number of services on offer and demand. These can be either software services or real-world services or both. Real-world services can be products or services exclusive of software services. The types of actors have different kinds of services on offer and demand, e.g. early Implementers of the DBE are mostly interested in software services. These can be seen as software oriented SMEs with a focus on new technologies in the field of software engineering and therefore join the DBE. Figure 4.3 summarizes the five types of actors and their ratio of services on offer and demand for software as well as real-world services. In the centre of the figure a view screenshots of the EvESimulator sketching SME networks while running in the simulation. Additionally, the existence of local service pools and user profiles is drafted.

At the current developmental state, the distinction between the types of actors and services is not implemented yet. However, the object oriented programming of the EvESimulator enables an easy adoption to the identified actor and service types.

4.2.3 Local Service Pools

The local service pool is a set of services registered at an Actor. This is where services that arrive, from other actors, are stored and it is the location from which service are deleted or migrated to partners. The service pool is the main source for one or more optimisation processes running at a time to fulfil the SMEs demand. The migration of services starts from the local service pool and the applied thinning algorithms are one of the most important factors for automated clustering. Therefore the following paragraphs give a brief overview of the internal processes and intercommunication related to the local service pool.

Services in the EvE- Environment migrate widely when they are used or limitedly avoid deletion. Migration in this context means that an exact copy of the service, clone, is made and afterwards transmitted from one SME to one or many collaborating SMEs, depending on the calculated migration probability. Triggers for migration events are 1) the service pool exchange, 2) the service pool thinning processes through *shrinking* algorithms and 3) the continuing optimisation processes of the service pools by the GA or other optimisation outputs.

The migration probability of one connection is the sum of the strength of all outgoing links divided by the strength of the considered connection. Important thereby is that always only the migration probability of one network can be calculated, e.g.

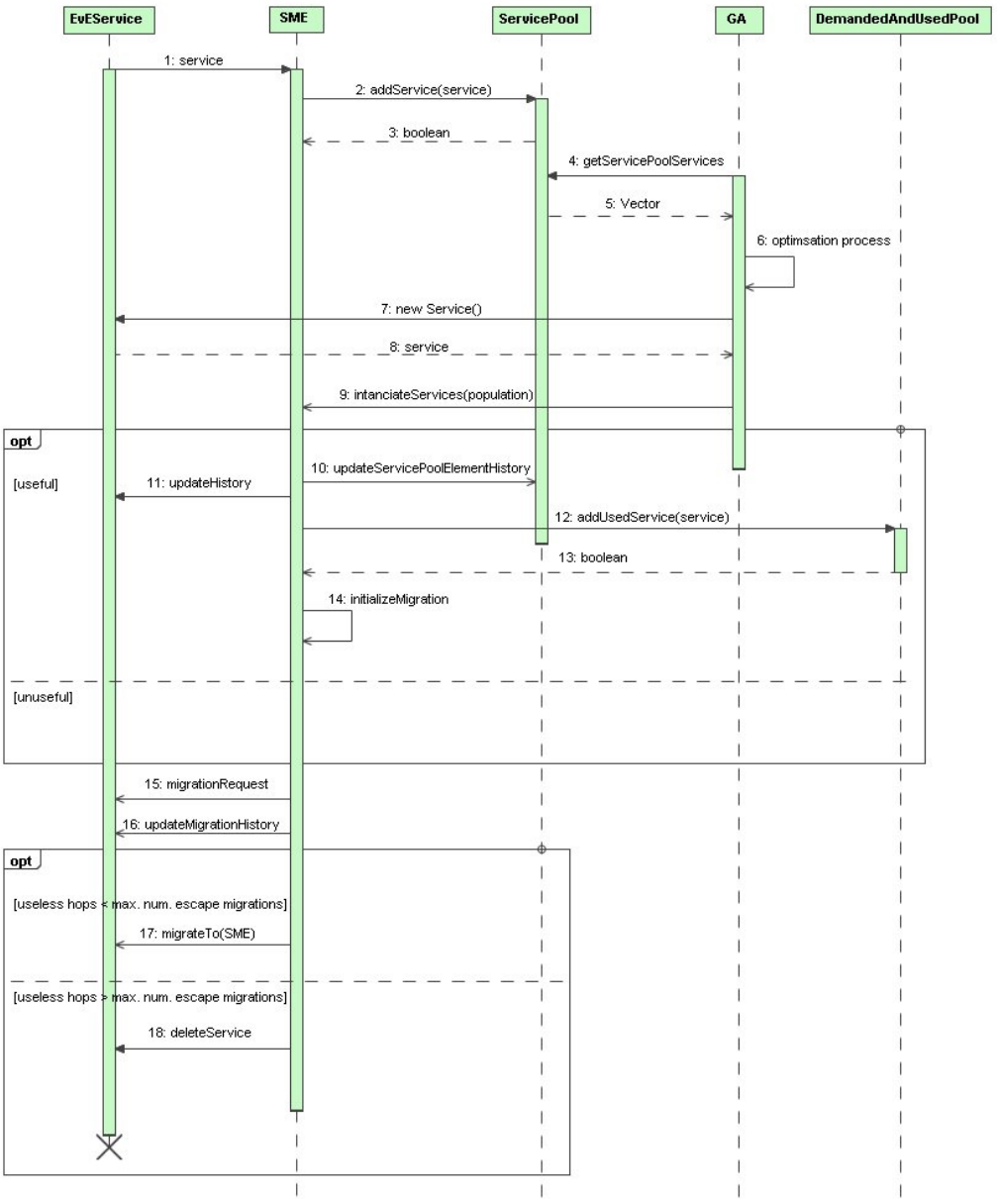


Figure 4.4: UML Sequence Diagram of a migration process implementation in the EvESimulator.

the migration probability of a link of the EvE network. Therefore, it is possible that a service is send two or three times to the same partner caused by different migration events. Many successful service migrations increase the connection strength and therefore the probability of future migrations. Successful means in this case that the migrated service is also useful for the target SME. For clarification of this complex procedure an example where a service migrates from SME1 to SME2 is given here. After a given time yet another service migration event is triggered. For following the procedure see also the collaboration diagram in Figure 4.4.

1. First of all, SME1 triggers a migration event to transmit the service to SME2.
2. After the arrival at SME2 the service will be added to the local service pool of this agent.
3. The service obtains a request to migrate from a running thinning process. Here we can distinguish two cases:
   (a) The service was chosen one or more times by the GA, while located at SME2, and was marked as useful. Useful in this context means that the service was used in, at least one, optimisation process as the best service in the last population. Beside the best fitness in the population the service combination has to be better than a service used previously to satisfy the request. In that case, the services history updated and subsequently distributed to one or more connected SMEs. Overall, the service has avoided deletion without being in danger of extinction.
   (b) The second possibility is that the service was useless for the SME. That means either that the GA chose the service one or more times without being useful or that, the optimisation process at SME2 never chose the service to fulfil a demand. If so, first the services history is updated. Second, in the case, that the number of hops without beeing used (useless hops), determined through the services history entries, is greater than the maximum of escape migrations, allowed for a service, the service will be deleted. Otherwise, the service lost only one of its “life points”. Therefore, the considered service has successfully avoided deletion as well.

Service Pool Thinning

The service pool thinning approaches was intended to optimise the loacal service pools so that a preselection of probably useful services for a certain SME is stored for future optimisations. We can also think of an extension of this approach with a search capability through the network if the needed services can not be found in the local service pool. At present two different approaches are implemented.

The first, but most complex, service pool thinning approach is the *shrinkByUseage* algorithm. This algorithm uses the history, stored in the local service pool elements of each SME, to decide whether a service should migrate, be deleted or hold. The decision bases, on the one hand, on the time step when the services was added to the local service pool and on the other hand on its usefulness, representing how often the service was part of an chosen service combination at a certain SME. For example, the oldest and less often used services are the candidates for the thinning process. That could mean that e.g. in the case of a service pool composed of eighty services and a *Shrink Factor* of ten percent, the four oldest as well as the four services that were the least useful, are removed. Removed in this context means that either the services are deleted or a migration event is triggered, to avoid deletion.

The second approach is the *Shrink By Profile* algorithm. To clarify this process here a short example: In the local service pool, ten services are stored. The system triggers a *Shrink By Profile* event. Now, each service, stored in the local service pool, will be compared against each demanded service. Strictly speaking, the attributes of a stored service are compared to the attributes of the demanded service. The result of this comparison process is a specific fitness value for each local service pool element. Therefore, the services with the worst fitness will either be deleted or send to connected SMEs.

The first simulation results for comparing these two approaches and evaluating their impact on clustering can be found in Section 5.2 and Section 5.3.

4.2.4 Interaction Patterns

The following paragraphs are mostly an outcome of the cooperation with partners from the business group, where many partners, especially Elmar Husmann (IBM) met with STU for working on lifelike simulation cases. In order to model as close to reality as possible, common business cases had to be identified. Small clusters of SMEs were discussed based on the declared intention of information sharing between

SMEs. As this intention can not be assumed without restrictions, the EvESimulator should show the benefits of information exchange and collaboration by showing concrete use cases in new regions.

Figure 4.5 summarizes the four identified interaction patterns of SMEs within the DBE. These are not exhaustive but reflect also clustering issues already outlined in [7]. By configuring simulations along these interaction patterns, the benefit of an open and self organized network, based on information exchange with suppliers as well as business rivals, could be shown. In the following the interaction patterns and corresponding questions are explained (see also Figure 4.5):

1. *Which SW services could enhance my current set-up:* The SME B is looking for some SW services which can enhance its business activities with D, E, and F. DBE provides, for example, an interface for sharing calender information between the partners to synchronize their activities.

2. *Which SW services are used by similar companies:* DBE SW services are already in use. A Peer comparison with other similar companies could enhance the performance by the application of new DBE services used by others automatic distribution of best practices.

3. *Which alternative BIZ partners are there in the DBE:* SME G provides similar products or services as E, and F. In the case of bottlenecks, SME B can search for alternative business partners along semantically described profiles. Additionally, the integration of the SW services is easier because both are using DBE frameworks.

4. *Which additional services in the DBE can broaden my offering and my business processes, respectively:* Networks and DBE services are set up. Now the information sharing can also broaden the product range of a SME.

Especially SMEs often benefit from keeping information about their products and processes private. One goal of the EvESimulator is to overcome the issue of information exchange in identifying the benefits of collaboration within a sector but also across sectors.

4.2.5 Social Networks

As already mentioned in Section 3.3.5 the visualisation and utilisation of social networks within the EvESimulator was one of the most recent ideas regarding dis-

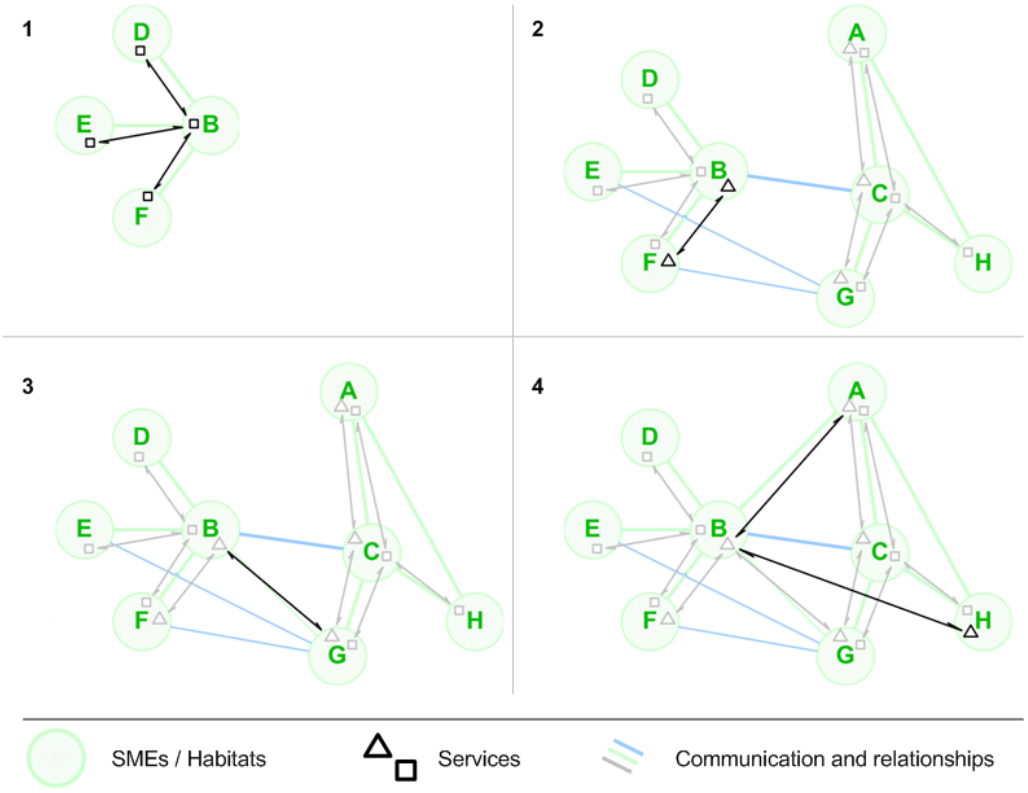


Figure 4.5: Interaction Patterns.

tributed fitness. For that reason, the initially planned functionality of building up networks by random or by certain topologies was extended. We can include now also findings from social science analysis and try to do approximations of relationships and also simulations on abstractions of existing social networks. As a matter of course social networks and the data that can be gathered from social network analysis is not sufficient for a full network analysis that can be adapted directly to a digital ecosystem. Additionally, we have to make assumptions in the following that are sometimes vague but the EvESimulator should start to converge the reality of social networks with network based research on P2P, evolutionary environments and self-organising networks.

Social relationships are assumed as the base for willingness of information exchange. The following table shows a possible mapping of trust with types of social contact. At the first sight this a good best guess but working further on the EvESimulator these mappings have to be underpinned by questionnaires as well as how to make an distributed approach more applicable to reality (see also [18]).

6	Personal contact	Trust
5	Sharing of resources	
4	Information sharing	
3	Partaking in projects	
2	Memberships in bodies	I don't know
1	I know them	
0	I don't know them	
-1	I don't trust them	Mistrust

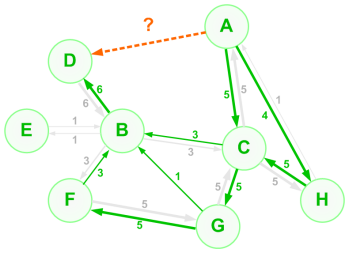


Table 4.1: Mapping of personal relationships between SMEs and their level of trust as well as a sketch of an SME network of trust.

In the EvESimulator, the networks of personal relationships and trust are considered as one of the possible decision criteria for choosing service combinations after comparing and assessing the attributes of the request with the attributes of the service combinations. On the basis of already connected nodes within the network, a distance function can be applied and the trust level of every node can be calculated using the Dijkstra Algorithm. The pseudocode of this algorithm can be sketched as follows:

```
1  function Dijkstra(G, w, s)
```

```
 2     for each vertex v in V[G] // Initializations
 3           d[v] := infinity
 4           previous[v] := undefined
 5     d[s] := 0
 6     S := empty set
 7     Q := V[G]
 8     while Q is not an empty set  // The algorithm itself
 9           u := Extract_Min(Q)
10           S := S union {u}
11           for each edge (u,v) outgoing from u
12                 if d[v] > d[u] + w(u,v)  // Relax (u,v)
13                       d[v] := d[u] + w(u,v)
14                       previous[v] := u
```

Additional information about implementing Dijkstra can be found in [19] and [20]. The Dijkstra algorithm enables us to calculate the value of direct relationship between two nodes in the network, no matter if they have already a connection or not. To show the application of Dijkstra in the EvESimulator, here a short example:

On the right hand side of Table 4.1 a small network of SMEs is drafted. Lets assume that SME A wants to buy a product of SME D. Although they don't know each other, SME A wants to consider a approximation of the level of trust it can expect when dealing with SME D. Therefore, the shortest path from A to B has to be found. With respect to this example, SME A already dealt with SME C, SME C with SME B and SME B finally dealt with SME D. The assumed level of trust implemented as a modified Dijkstra in the EvESimulator is calculated as:

$$\text{trust}_{(\overline{AD})} = \max \cdot \exp\left(\sum_{i \in I} \log\left(\frac{\text{trust}_{(i)}}{k}\right)\right)$$

Where I is the shortest path.

$$\text{trust}_{(\overline{AD})} = \left(\frac{\text{trust}_{(\overline{AC})}}{\max} \cdot \frac{\text{trust}_{(\overline{CB})}}{\max} \cdot \frac{\text{trust}_{(\overline{BD})}}{\max}\right) \cdot \max = \left(\frac{5}{6} \cdot \frac{3}{6} \cdot \frac{6}{6}\right) \cdot 6 = 2.5$$

This was an example of the current implementation of Dijkstra in the EvESimulator in order to show the implementation status. An exchange of this implementation with another for a certain simulation case is straight forward and easy implementable.

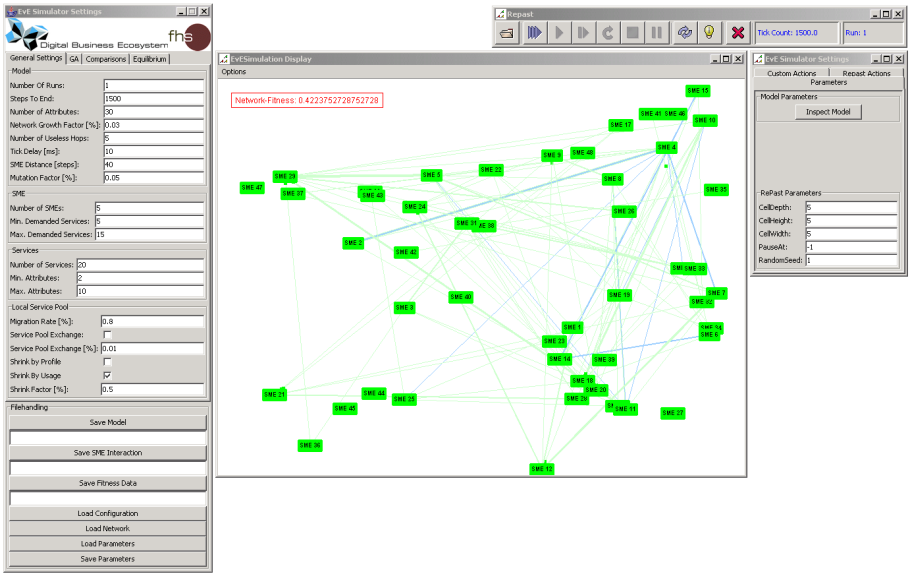


Figure 4.6: EvESimulator Graphical User Interface.

4.3 EvESimulator Core Features

After outlining the main characteristics of the implemented model the following sections give a summary of the implemented features and especially configuration capabilities of the current version of the EvESimulator. We try to describe the features along the graphical user interface (Figure 4.6) so that this acts also as a kind of user documentation for the EvESimulator. For implementation details see the javadoc section at [9].

4.3.1 Standard Features

The graphical user interface of the EvESimulator consists mainly of four windows. Three of them are coming with repast and the fourth, additional settings window was developed from scratch (see Section 4.3.2). The following section only describes the main features of repast which can easily be used. For detailed documentation on repast as well as mailing lists etc. see also [13].

In order to run a simulation, the standard repast toolbar (right upper corner in Figure 4.6) can be used. Beside the start button, stepping per tick, initialising the model, stop, pause, refreshing the setup and exit are implemented. Although the *Start Multi-Run* and *Load Model* features are not deactivated a adapted version

of these features is implemented in the *EvE Simulator Settings* window and also recommended to use for running DBE simulations. The *View Settings* button will display the repast model *Settings Panel* if it is hidden or closed.

Although, the configuration of the Simulator would be possible also through the *Settings Panel* of repast (right side of Figure 4.3.2) the customization of this element would have been not effective for the various configuration capabilities of the envisaged EvESimulator. After eliminating all the configuration parameters which can be done in the later described *EvE Simulator Settings* window, core features of repast are left there enhancing the specifically implemented features. One of the most recognizable features is taking a snapshot of the running simulation or making a movie of the simulated network on the fly. Consequently, presentations of the network simulations can be supported by a movie without having the tool installed and configured.

For visualising the capabilities of the Evolutionary Environment, the actors, services as well as the whole network topology can be displayed through the *EvESimulation Display* (center of Figure 4.6). Beside the visualisation of the network a label for displaying the *Network-Fitness* was introduced. The calculation of this fitness is calculated by adding all the fitness values of the SMEs. This algorithm can be changed in future for example by calculating the fitness of the network by ICLs global optimisation equation (see [21]). The customized features of the display window is extended in Section 4.3.4.

4.3.2 EvESimulator Configuration

Figure 4.7 shows the *EVE Simulator Settings* window which acts as the main configuration window for the EvESimulator. It enables the user to configure the main characteristics of an simulation without changing the source code. That is particularly important for non-technically experienced groups with a broad knowledge in networks or social interaction.

The *EvE Simulator Settings* is structured in four configuration tabs, namely *General Settings*, *GA*, *Comparisons* and *Equilibrium*. Moreover, at the bottom of the window the import and export features are situated which are described in Section 4.3.3. In the following the structure and features of the *EvE Simulator Settings* window are outlined according to Figure 4.7.

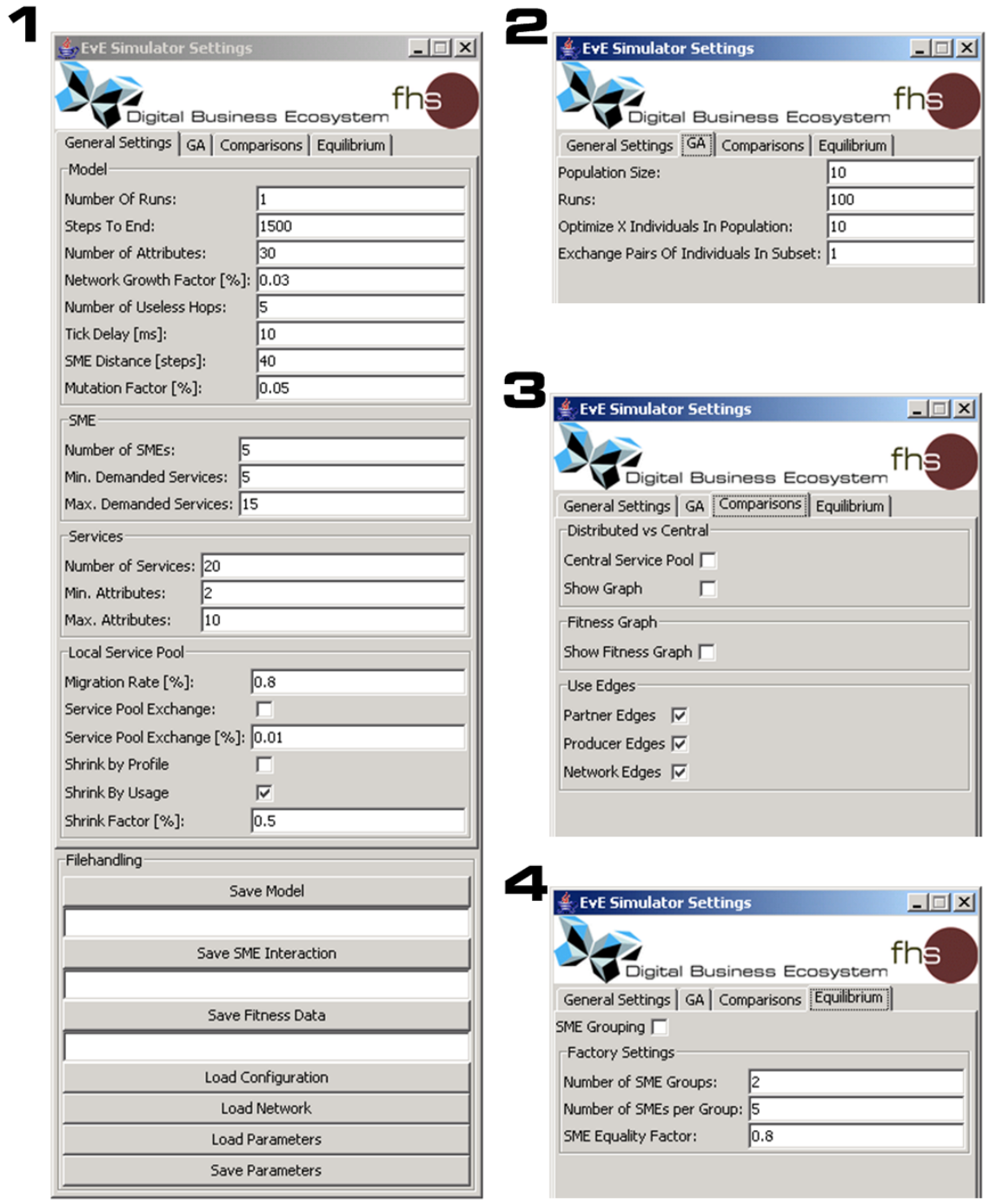


Figure 4.7: Customized configuration window of the EvESimulator

General Settings

In order to run more than one simulation in a sequence *Number Of Runs* and the needed *Steps To End* of each run can be set in the Model group. At the end of each run, statistics are stored in the predefined files and the next run is started automatically. Consequently, the simulation can be executed during over a long time period without any user interaction, when more CPU power is beeing needed to run the simulation. The *Number of Attributes* is one of the most influencing parameters. It represents the heterogeneousness of the system. Furthermore, it can be seen as the available diversity of the SBVR vocabulary. When the simulation is initialized a attribute pool is generated randomly with the configured number of different attributes. The more attributes the less probable is the perfect match of a optimized service in comparison to the demand for example. The attribute pool, as it is implemented at the moment, is static. That means that after the initial creation only the randomly generated attributes are available within the system. For further extensions as regards self-organisation we can think of applying mutation to the attribute pool while the simulation is running. The *Network Growth Factor* indicates in percent the growth of the network per tick. For simulations of static networks this factor can be set also to zero percent. Next, the *Number of Useless Hops* or escape counter, respectively, indicates when the clone of a service is removed from the network after being not useful at several SMEs. Further information on this procedure can be found in Section 5.2. Because of the fact that the simulator can also be used for presentations, the simulation can be slowed down by configuring the *Tick Delay* of each step in milliseconds. Analogical, the *SME Distance* offers a more lively presentation of the simulation. In order to visualize the movement of the services from one node of the network to another, the services are not copied directly within one step to the target of the migration, but on the contrary are sent to the edge and migrate step by step. The number of these steps can be configured as the *SME Distance* and heads in a visualisation of the movement of services from one SME to another. The last parameter in the model group is an general *Mutation Factor*. This parameter is currently not used but can be easily built in every component which should mutate over time, e.g. mutation of attributes.

In the SME group, three basic configurations for SMEs can be set: The initial *Number of SMEs* in the network, the *Minimum of Demanded Services* as well as the *Maximum of Demanded Services*. Depending on these parameters the initial set of SMEs are initialized and the demanded services are generated by composing

services out of the previously generated attribute pool. The configured number of SMEs applies only if the SME grouping mechanism in the *Equilibrium* tab is disabled. Details of this feature are outlined later in this section.

Analogous to the SME group, the *Number of Services* per SME as well as *Minimum of Attributes* per service and *Maximum of Attributes* per service can be configured. Each SME gets the same number of offered services at the beginning. These services represent the total offer of the SME and are cloned also to the local service pool of this SME for further dissemination to potential users.

The last main group in the *General Settings* tab is the Local Service Pool configuration. The *Migration Rate* parameter is not used at the moment. This first listed rate can be used if additional migrations are necessary in future. We introduced this parameter for trying out bootstrapping approaches in future. Unfortunately, the limited resources for the implementation of the EvESimulator have not allowed us to work extensively on simulation cases yet because the set up of the framework and the model was considered as more important. Though, the current EvE service implementation was intended to support all envisaged exchanges of services or service information, respectively, the current version only facilitates the exchange of service pools with other SMEs. Since the EvESimulator should include a possibility for simulating the EvE also in its current status, and because of the fact that service pool exchanges could be a good approach for bootstrapping the system, the *Service Pool Exchange* check box starts a regular service pool exchange with the configured percentage of the network. Further information on bootstrapping the DBE can be found in [7]. The percentage of service pool exchanges represents the mean percentage of all network nodes reachable at this moment. Together with the setting of the scheduling settings of repast it can be adapted for any other bootstrapping mechanism envisaged in future. The exchange of the service pool, a simple but significant procedure, combined with the thinning process is necessary to ensure that also, beside the exchange of information among partners, a constant exchange of information between nonpartners is taking place. The result of this interaction is that all service pools of in this process involved SMEs are identical. Consequently, exchanges of local service pools have to be used carefully. The last parameters in the *General Settings* tab are the most crucial ones for building up network clusters and therefore, supporting the self-organisation facilities of the EvE. The underlying mechanisms are outlined in Subsection 4.2.3.

GA

As shown in Figure 4.7 (2) the basic parameters of the genetic algorithm (GA) can be set as well out of the *EvE Simulator Settings* window. The current implementation of the GA based on a joint implementation of UBham and STU during a workshop in 2005. Because of the circumstances outlined in Section 3.3.1 we have no up to date implementation of an customized GA from other partners. Therefore, a few modifications were done and the existing structure of the first joint GA implementation was used for the EvESimulator. In the *GA* tab, the basic configuration parameters can be set. These are the *Population Size*, the number of *Runs* as well as the how many individuals in each population are considered for optimization. In the currently implemented version of the GA, a randomly selected number of services (*Optimize X Individuals In Population*) are taken for applying crossover. After selecting this list, the least fittest services in this list are replaced with crossover product of the fittest pairs of individuals. The *Exchange Pairs Of Individuals In Subset* parameter defines how many pairs of individuals in the selected subset are considered for crossover. Subsequently, the optimised subset replaces the original selected services in the population.

The implemented mechanism for avoiding bloat is processed accordingly to the optimization algorithm. Consequently, *Optimize X Individuals In Population* parameter indicates again the number of randomly selected services out of the population. Here the shortest individuals in terms of number of attributes are selected, mutated (point mutation of the attributes) and replace the longest individuals in the subset, respectively. Finally, again the resected subset of the population is replaced by the modified one. The number of mutated attribute pairs is configured again through the the *Exchange Pairs Of Individuals In Subset* parameter. For further information take also a look at the Javadoc section at [9].

Comparisons

This tab contains mainly visualisation and comparison features. First, the evaluation of a *Central Service Pool* parallel to the distributed evaluation can be configured and presented by enabling the *Show Graph* checkbox. Furthermore, the *Show Fitness Graph* checkbox results in an additional graph which shows the gradient of the network fitness at a glance.

The most important configuration capability in the *Comparisons* tab is the setting of the *Edges* in the network. Particularly, the migration and self-optimisation

of the network depends strongly on the networks used for network building. By enabling or disabling certain edges, these are not only invisible, but also not available for the migration process and consequently the establishment of new collaborations. The detailed effects of the enabling certain network edges will be elaborated in future research.

Equilibrium

In order to show the proper mode of self-organisation operation as well as cluster building a factory was implemented in the simulator for producing networks with strictly defined characteristics. By enabling *SME Grouping* the *Factory Settings* are taken for the initialisation of a network with a defined *Number of SME Groups*, each with the defined *Number of SMEs per Group*. Even though, the grouping is done by construct most of the services and features randomly, the *SME Equality Factor* indicated how many of the demanded services within a group are equal. That leads to a predictable optimal clustering within these SMEs, where different setting and algorithms can be tried out for speeding up the cluster building process.

Additionally, the attributes demanded by services (potential requests) of one group are taken to compose offered services of the other groups. That leads to a customer - provider relationship between certain SMEs in the network which is also almost predictable. For more details about cluster building in the simulations see also Section 5.2.

4.3.3 Import and Export Capabilities

In order to persist the states of the EvESimulator, an XML based import and export facility was implemented. The EvESimulator should enable all disciplines of sciences to cooperate on a common framework. Therefore, import and export functionalities have to be usable for technically not experienced as well as technically experienced people.

The social network analysis within the DBE currently uses a SME table for retaining the relationships of SMEs. Rows as well as columns hold the names of the SMEs. The type of relationship (see also Table 4.1) is represented as the value in the intersection of axes. Furthermore, the types of interaction as in Table 4.1 can be imported as separate files where the name of the file indicates the type of relationship and the intersection of axes the weight of the interaction. As to provide

a common import from a broad range of spreadsheet software the import files for the EvESimulator have to be CSV (Comma Separated Values) using a semicolon for separation (see Figure 4.7 - *Load Network* button).

Import of SME data and their network is not sufficient in order to guarantee simulations with a consistent and reproducible state of the EvESimulator. Therefore, we developed a XML-based import and export functionality which enables a full snapshot of the simulation state. Validity checks using XSD as well as presentation of SMEs and services as HTML using XSLT is also available. An example of the XML files used for storing the simulation stage are available in the Appendix. The *Save Model* button in Figure 4.7 (1) opens a file dialog where an XML file can be chosen for storing the snapshot. When the STOP button is pressed the current simulation status is stored in the previously selected file. For continuing a simulation the same XML file can be chosen pressing the *Load Configuration* button.

Furthermore, *Save SME Interaction* and *Save Fitness Data* log the degree of interactions and progression of the calculated network fitness in CSV files. Beside these log files a XML based configuration proves useful for such a broad range of parameters in the EvESimulatior. Therefore, the whole parameter set can be persisted and reloaded from an XML file (*Load Parameters*, *Save Parameters*). These parameter files can also easily exchanged between partners for exchanging best practices for simulations or for an easy configuration during a presentation.

4.3.4 Web-Based Visualisation and User Interaction

Beside the visualisation of networks in the *EvESimulation Display* we included additional features integrated in the repast standard library. The intention was to modify the standard library of repast as little as possible but nevertheless some modifications had to be done. These modifications lead to three additinal "on-the-fly" interactions with the *EvESimulation Display*: 1) adding new SMEs on click, 2) adding new services on click and 3) showing the current status of an SME in a webbrowser. The following paragraphs elaborate on these new features.

First, with a simple right-click in the *EvESimulation Display* panel, a new SME is generated. It follows the configuration parameters set previously for the generation of the other SMEs. This event can be couple to a observation and reclustering of SMEs and therefore simulate the dynamic behaviour at any given time step. Because of the limited resources a concrete simulation case was not implemented yet.

By analogy, the right-click on a certain SME results in adding a new service to

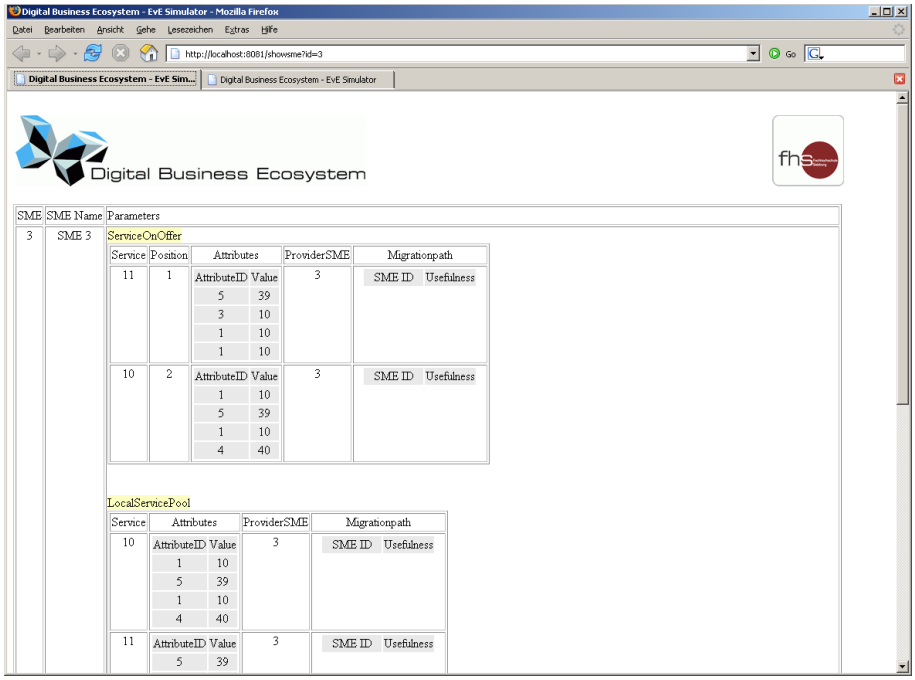


Figure 4.8: Screenshot of an SMEs XML-based representation by a webbrowser.

the list of offered services and consequently, also to the local service pool. From then on the service underlies the migration mechanisms of the system. The new service is colored differently so that the migration can be observed in a presentation. Moreover, statistics can be implemented in future to observe the life cycle of an service, introduced in a stable and clustered community of SMEs.

Third, the XML import and export utilities were utilised to present the snapshot of an SME in an attractive way. Thus, the left-click on a SME opens a browser window and represents the XML based representation of this SME through availing XSL transformations (see Figure 4.8). The technical realisation was done by using a *Jetty* webserver. For more details on *Jetty* see also [22] or the javadoc section at [9].

5 EvESimulator Preliminary Simulation Results

The following three implemented simulation cases represent the first results of utilising the EvESimulator framework. Although, the simulations present only preliminary results which need further refinement, the data is very promising for further research on Digital Business Ecosystems. For each of the simulation cases, an XML based parameter file can be found in the appendix (Critical Mass Assessment 5.1, Clustering 5.2, Usage Based Clustering 5.3) which can be easily imported into the EvESimulator. Consequently, all the simulations can be extended and redone by using the data presented in this document. The simulation cases presented here took from six up to twelve hours, each.

5.1 Critical Mass Assessment

While working on Digital Business Ecosystems and on their simulation, one aspect was omnipresent: what is the critical mass of individuals in a system like that. In [8] the concept of business ecosystems was, beside others, stated as follows:

> [...] As in natural ecosystems, the digital species should have enough individuals to survive and the digital ecosystems should be populated by a sufficient number of species (a critical mass of species) to be appealing for the market and continue to exist. [...]

To anticipate the following section about critical mass assessment: it will also NOT give an answer to this issue. Nevertheless, we tried to implement in the EvESimulator most of the features which are needed to do approximations for an calculation of critical masses. Moreover, we did a first simulation case where a distributed optimisation approach has been compared with a centralized optimisation.

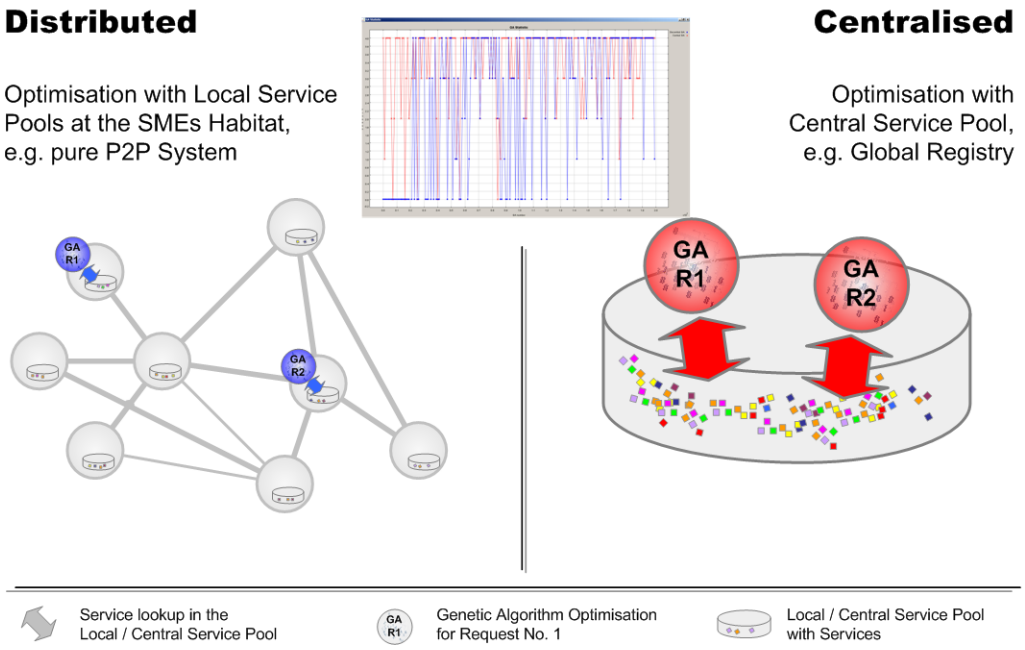


Figure 5.1: Concept of distributed versus centralized optimization

Figure 5.1 sketches the main ideas behind the implemented simulation case. On the right hand side, a central registry is shown. In this case the optimisation is triggered with knowledge of all the services available in the ecosystem. This approach is comparable with a search in the DBEs Semantic Registry for example.

The left hand side of Figure 5.1 sketches a distributed optimisation facilitating local service pools. It bases on the assumption that the self-organising mechanisms of the EvE will distribute services over the network according to their probable usage in future. Therefore, the services in the local service pools prove to be useful for the demand of the SME. On one hand this pre-selection speeds up the optimisation process because the search space is smaller and the services are already pre-selected, on the other hand there is no knowledge about all available services and consequently, the available solutions are limited to the services in the local service pool. Moreover, a fast change in the demand of a SME can lead to solutions with low fitness.

Summarizing the *Critical Mass Assessment* simulation case, we configured 4 networks with varying numbers of SMEs and attributes. In order to evaluate the preliminary outputs, we documented two hypotheses:

1. Hypothesis: The results of the centralized optimisation will be better in the bootstrapping phase of the system. After the bootstrapping, the distributed

optimisation outperforms the centralized approach because the optimisation is preorganised and thus, operates on a smaller searchspace.

2. Hypothesis: The number of attributes (diversity of services) as well as the number of SMEs in the network have a considerable influence on the simulation results.

For the parallel optimisations, the EvESimulators genetic algorithm was used. In order to provide comparable results, the same configuration was used for both, centralized and distributed optimisation. The detailed, XML based configuration can be found in the Appendix.

Results

Figure 5.2 summarizes the first results for the *Critical Mass Assessment*. For each graph, 50 simulation runs were performed and consequently, each point in the graphs represents the mean value of 50 datapoints. The simulations were carried out with the same parameter set except two observed variables. Graph (1) and (2) show the results for a network with 10 SMEs, whereas (3) and (4) show the respective results of a network of 50 SMEs. In the simulation presented by graph (1) and (3), the overall system was set up with 20 attributes for describing services. The number of attributes in the systems represented by graph (2) and (4) was set at 40.

The presented results underline *Hypothesis 1* as all graphs in Figure 5.2 indicate the better performance of the distributed optimisation compared with the centralized after a number of iterations. Contrary, the fitness of the result set presented by graph (1) drops decreases in the last quarter of optimisations. At this point additional simulations have to be done in order to identify the reasons for that behavior. After running the first simulation cases by the use of the EvESimulator, the local service pool thinning was already recognized. Certain settings resulted in a over-reduction of the local service pool and therefore not enough services were available to build up satisfying solutions. That could be one reason for the decrease of fitness at the end of graph (1), although this is only an unproven assumption.

Hypothesis 2 was underlined as regards the impact of the number of attributes. Whereas, the simulation cases (1) and (3) indicate only slightly better results of the distributed optimisation, in (2) and (4) the distributed optimisation results in considerably better solutions. The considerable difference was achived by only doubling the number of attributes from 20 to 40.

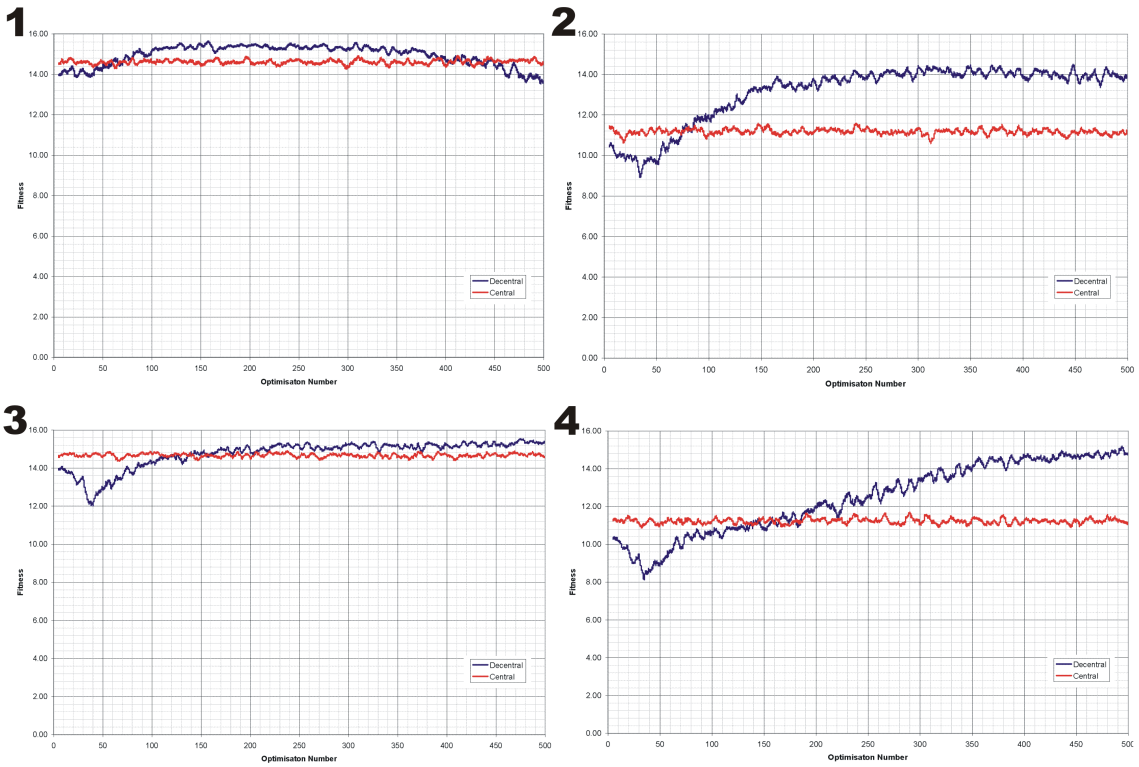


Figure 5.2: First critical mass simulation results (1) network of 10 SMEs and 20 different attributes in the system, (2) network of 10 SMEs and 40 different attributes in the system, (3) network of 50 SMEs and 20 different attributes in the system, (4) network of 50 SMEs and 40 different attributes in the system.

The second part of *Hypothesis 2* included the impact of the number of SMEs on the result set. Although, the adding of SMEs from 10 to 50 resulted in a longer self-organisation phase, the influence as regards fitness was not apparent.

5.2 Clustering

In order to test the capabilities of the EvESimulator and to show what are the main advantages of the self-organising network we worked out typical interaction patterns of the SME networks in the future DBE. The four identified business interaction patterns regarding self-organisation were already shown in the Figure 4.5.

When taking a look at the self-organisation of such a real-world scenario it has to be differentiated between 1) real-world network evolution and 2) the EvE network evolution. The color coding in the simulation visualizes the two different networks for better traceability. Real-world network evolution results out of communication and networking of people in a certain area. For example in the context of a public event (conferences, meetings, etc.) new information channels are created and new personal contacts are set up which can be utilised for future business activities. The EvE network evolution and self-organisation relies on the behavior of requesting and instanciating services in the EvE network. New network connections and edges between network nodes are set up because the system identifies, for example, that a service was useful at another node as well and consequently a direct edge is created to accelerate the information exchange between the two nodes. Beside the seven types of interaction extracted from the collaboration with the Censis (Table 4.1) the EvE network is supposed to self-organise artificially. One possible approach for the EvE evolution and the resulting self-organised network building is described in the following paragraphs.

Migration

The migration of services which is one of the key concepts of the EvE depends strongly on the outgoing edges of a node (SME representation) in the network. As a consequence of the twofold network evolution the test runs should be possible first, considering only the real-world network connectivity, second, only the EvE network connectivity and third, considering both. The real-world connections are representing social contacts and have a maximum value of 6 which stands for personal contact. In the the EvE network which builds up during usage of the system

the strength of the edges are not limited naturally, e.g. each useful migration leads to an increase in the network connection. Therefore, the maximum value for calculating the migration probability is set to the maximum value of all outgoing edges at this moment in time.

For clarification here an example: When the migration event is triggered for a certain service the service is migrating depending on the strenght of the edges. We calculate now the migration probability of three outgoing edges a, b, c with strenghts a(2), b(5) and c(1). The migration probability of a(2) in the real-world network would be 2/6 because the maximum value for these edges is set with 6. On the contrary the migration probability in the EvE network would be 2/5 because the maximum value of all outgoing edges is 5. This calculation is necessary because there is no global information available in a pure P2P network and therefore the maximum value of all edges in the network is not available.

Trigger of Migration Events

According to [7] EvEServices migrate only 1) when used or 2) if they want to avoid deletion (when excluding the DIS). Firstly, in the EvESimulator using or instanciating a service results out of the following activities:

1. Demanded service at a SME node is taken as a request for a new optimisation
2. GA thread is started to search the best fitting combination (with respect to the demanded service in 1) of services from the local service pool
3. GA ends with the last and fittest population of service combinations
4. SME decides which service should be instantiated (in this setting always the fittest of the last population is chosen)
5. All services in the chosen combination are marked as useful for the requesting SME (history update)
6. For all services in the chosen combination a migration event is triggered
7. A new service is assembled and put to the UsedAndDemanded services (see 4.2.1) as well as to the local service pool
8. A migration event is triggered for the new service

Secondly, services have a limited time span at a SME. Regularly, the service pool is thined out by checking if services are potentially useful for the SME. Useful in this context means that the service was used in a GA run and in the last population of such a GA run, respectively, as well as if the service fits to the profile of the SME. As the profile of an SME in the simulation is equivalent to the set of demanded services each service is assessed relative to the demanded services of the SME. The less fittest individuals are in danger of deletion. If they did not migrate the maximum of hops, a migration is triggered for these services. If they reached already the maximum of hops, they are deleted. The migration itself relies again on the migration process described above.

Evolution of Network Edges

As there are two types of networks (real-world and the EvE) there are also two different types for the the increasing and decreasing of the edge strenghts. First, the real-world scenario is built up on social networks. By utilising an adapted Dijkstra algorithm the level of personal relationships can be calculated for each pair of SME nodes which are connected to the network and not directly connected to each other. The increasing and decreasing for social-network connections is not defined yet. Consequently, only new edges between nodes can be created by using the adapted Dijkstra algorithm in the EvESimulator (see also Section 4.2.5).

When a new service is instantiated at a SME and put into the DemandedAndUsed and in the *Local Service Pool*, new edges are created or existing ones are strengthened. In this case a new edge is created to the providers of each service in the combination (light green) as well as to all SMEs where the service history indicates "useful" (light blue). If there is already a edge to one of these SMEs, the value of the extisting connection increments by the value 1. As a consequence the EvE network evolution should end up with clusters of companies which use similar services (light blue) and chains of SMEs which use services of their providers (light green).

Moreover, the productivity of a migration to a SME is assessed. When a service migrates to a SME it can:

1. be used during an active request (GA run) and instantiated together with other services

2. be useless for a SME and at some point in time endangered by deletion

In the first case the service gets the permission to migrate. Consequently, the service is cloned and the history of the service is extended by an entry SME name, useful = true. Therefore the last migration of the service was useful and the correspondend edge is incremented by 1. The second case indicates that the last migration was not useful. Hence, the weight of the last correspondend edge is decremented by 1.

Simulation Case

In order to test if the interaction patterns will set up automatically as anticipated to test the functionality of the system, a test-set was created with a number of SMEs with similar demand. Therefore a XML configurable data generation was implemented. For the configuration of SME groups see Section 4.3.2. In this first simulation case for clustering two groups of SMEs with three SMEs per group were generated. The SME *Equality Factor* was set to 90 percent. Algorithm of choice for thinning out the local service pool was *ShrinkByProfile* (see also Section 4.2.3).

Results

The Habitat network was intended to allow spontaneous formations of communities via clustering within the network (see [7]). Consequently, in the DBE community several approaches were discussed but most of them were for a high number of services and SMEs. On the contrary, this first simulation case by was targeted on a small network with six SMEs in total. Therefore, we decided to use the *ShrinkByProfile* mechanism because it doesn't need a long bootstrapping period in order to offer good results (see Section 4.2.3). The intention was to split the network into SME clusters, created with similar demands (high *Equality Factor*). The assessment of the clustering behaviour had to be done manually by analysing the history of any used service of each SME. For clarification, Table 5.1 shows an example output of one simulation run.

All service in the simulation have a history of SMEs where they were useful as well. Such a *useful* credit is assigned to a service if a service was chosen for instantiation at a certain SME. Parallel to that credit all the SMEs where a service was useful are stored at the instantiating SME. Table 5.1 summarizes the usefulness of services per SME. Though, the considered Agent/SME is listed in the first column, the set of SMEs, where the services used by this particular SME were also useful, are listed in the *History* column of Table 5.1. By the aid of the given result table, the

CONSIDERED AGENT	HISTORY						
SME1	SME3	SME3	SME2	SME2	SME2	SME3	
SME2	SME1	SME1	SME1	SME1	SME1	SME3	SME1
SME3	SME1	SME2	SME2	SME2	SME1		
SME4							
SME5	SME3						
SME6	SME5	SME4	SME4	SME1			

Table 5.1: Example logfile of one simulation run in the clustering simulation case.

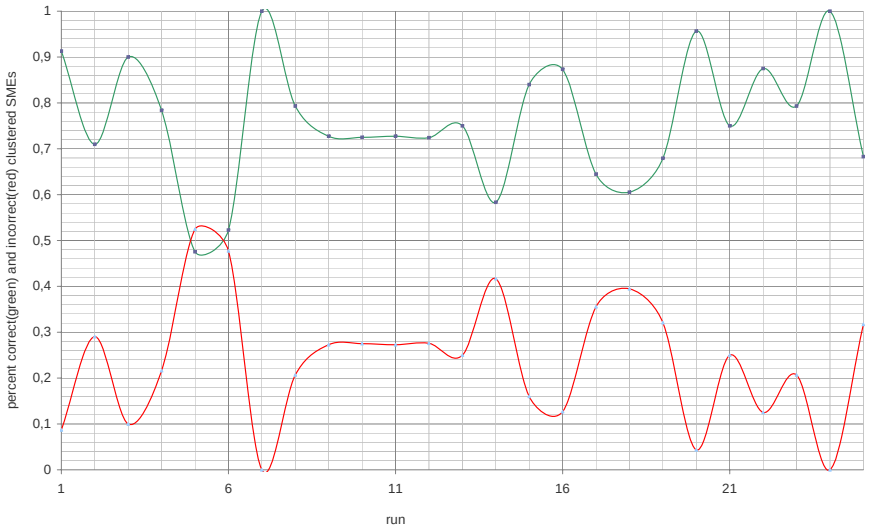


Figure 5.3: Evaluation graph of the clustering behavior of profile based clustering (green is correct, red is incorrect).

clusters can be easily observed. The first SME cluster is represented through SME 1-3. The remaining SMEs (4-6) belong to the second group. Consequently, services, used by SME1 are also useful at SME2 and SME3. However, as demonstrated by this example, a very high percentage of useful services, satisfying the needs of a certain SME, were also useful to other SMEs of the same group, resulting in SME clusters.

On the whole, the EvE network performed as expected, adapting and improving over time to reach the maximum performance level as well as to cluster SMEs with similar demand. Overall, after analysing 25 simulation runs, more than 90 percent of the evaluation runs demonstrated exceedingly clear the clustering effect (see Figure 5.3). Although, the anticipated clustering behaviour set up as expected for this example, further research on the most important factors of clustering, bootstrapping as well as authenticity of the model is necessary.

5.3 Usage Based Clustering

Out of the fact that the descriptive language of choice in the Evolutionary Environment is SBVR, these two issues are still strongly coupled. Although, the concept of the Evolutionary Environment is much more than SBVR matching, the complexity of matching SBVR models has some considerable drawbacks for several concepts in the EvE as well (for SBVR matching see [1]). Consequently, we focussed the third simulation case implementation on an EvE without model matching.

Clustering networks of SMEs and therefore, speeding up the information exchange and distribution of services is one of the core benefits of the upcoming EvE implementation. In [7], clustering mechanisms are described and simulated in detail and act as a basis for the EvE implementation. Furthermore, this clustering can be seen as the first self-organisation of the DBE network topology. Section 5.2 elaborates on the profile based clustering mechanisms, which, again, used the comparison of service descriptions relative to the demanded services.

In this case, the distributed optimisation, in the usage based clustering, uses exclusively the simulated user behavior for self-organisation. The optimisations, done by the genetic algorithm, simulate the users decision process ending up with the "best" service combination from the local service pool. This selection is the only parameter for the migration of services and consequently the clustering mechanisms.

Because of the limited resources for implementation, we did only some first simulation cases. These are limited to a few runs because of the considerable time needed for running the simulations. Figure 5.4 shows the output of the profile based clustering simulation. We triggered ten runs and show the percentage of correct clusterings equal to the one described in Section 5.2. Taking the same parameter set a second simulation was done by changing only the type of service pool thinning. Therefore, instead of the *Shrink By Profile*, the *Shrink by Usage* algorithm

was chosen (see 4.2.3). Consequently, instead of matching models, only the usage behaviour was taken to organise the topology of the network.

Although, this was the first trial of clustering only based on usage behavior, the graph in Figure 5.5 indicates that a clustering of SMEs could probably achieved also without the envisaged SBVR model comparison. Nevertheless, considerable research effort is needed in future to aquire concrete simulation results for the various EvE scenarios. Moreover, these simulations are only examples, which have to be adapted to the concrete contexts in which they should provide the first approximations for an introduction of an DBE.

Figure 5.4: Evaluation graph of the clustering behavior of *Profile Based Clustering*.

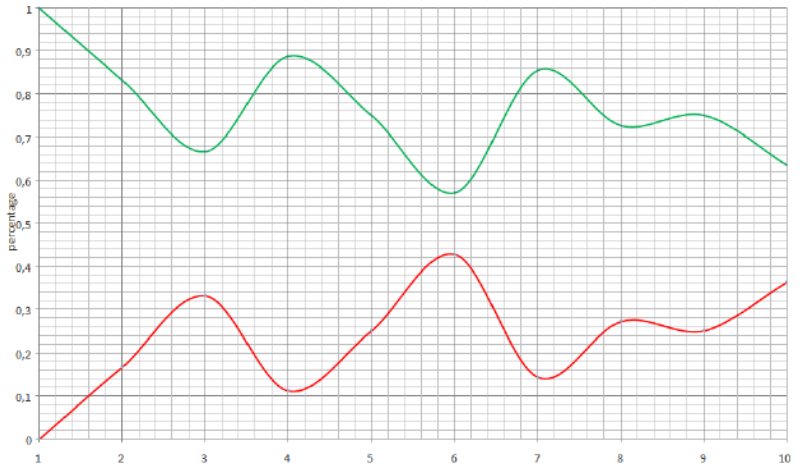


Figure 5.5: Evaluation graph of the clustering behavior of *User Based Clustering*.

6 Conclusion

In order for the DBE to be useful in the constantly changing market, the EvE service implementation outlines the way that interconnected but independent habitats are to be used to accommodate this change. To be useful for the common user of the system, it also makes use of SBVR, a natural language based business modelling system. Although still in the development stage, the EvE Network strives to be an independent, flexible, continually evolving, and practical system in the business world.

The Evolutionary Environment and the EvESimulator can be seen as the bridge between the areas of science, computing, business as well as social science and this is where the analogy of the natural ecosystem is the strongest. To keep the simulator as realistic as possible and while attaining the goal of speeding up the process of evolution, a number of tools were used to simulate the system. Repast and the Java Enterprise Simulator Program have shown to have good results in the simulation. Repast and the EvESimulator were to tackle the goal of having a system where the habitats remember past interactions with different habitats and services to continually improve the system in a smooth way.

As this is a large project involving several groups from the DBE consortium, the EvESimulator is still being developed. The realism of the simulator itself required the collaboration of almost all disciplines within the DBE consortium between 5 universities, namely GAs from UBham, Global Optimisation from ICL, Symbiosis and Competition from HWU, Social Networks, as well as the programming at the STU.

The *EveSimulator Implementation* section outlines some of the ways how the theoretical ideas behind the simulator were implemented. SBVR has a large role dealing with the interface with the user where it is then used to represent requests and solutions using the bitstrings. In this chapter, we define the roles of the actors involved, and how their interaction patterns, the social network, are to be assessed

by using Dijkstra's algorithm and by calculating the relationships using appropriate tables.

This chapter also explained the interface of the EvESimulator and how the system is to be used. The functionality of the graphical interface is mapped and how it can be configured.

In the *EvESimulator Preliminary Simulation Results* chapter, critical mass is defined as the number of species needed to allow the ecosystem to effectively be sustained. The EvESimulator is run to test the differences between a distributed optimisation approach and one with a centralised optimisation. Here we proposed and backed the idea of two hypotheses: First, we found that centralised optimisation is better in the first set of iterations but distributed optimisation outperforms this as the number of iterations go on. It is believed that the reason for this is because the distributed system is due to the fact that it is in a smaller search space. Next we found that the number of attributes in network has considerable influence on the simulation results. It takes slightly longer to get evolution going but better results.

We then dealt with the issue of interaction patterns. Namely we address the issue of clustering and the seven types of interactions of a social science group. The rules of migration are further discussed for the management of the service life-cycle. We also define usefulness and the evolution of network edges and their strength properties. Testing showed us that there is an obvious presence of clustering and that further research into it and bootstrapping will have to be done.

Lastly, usage based clustering is investigated and evaluated in comparison with other clustering methods. Here, only the user behavior was taken without considering the possibility of SBVR model matching. This was the first time that this was tested but shows promising results as clustering of SMEs could probably be achieved. We believe that considerable research is needed in the future to run more full tests to represent the dynamic EvE settings.

Bibliography

[1] Giulio Marcon, Hisanaga Okada, Thomas J. Heistracher, and Thomas Kurz. Report on Adaptive Service Generator. Deliverable D16.3 Digital Business Ecosystems Project, June 2006.

[2] Thomas Heistracher, Thomas Kurz, Giulio Marcon, and Claudius Masuch. Evolutionary Environment Service Implementation Technical Documentation. Deliverable D9.3 Digital Business Ecosystems Project, April 2006.

[3] Thomas Heistracher, Thomas Kurz, Giulio Marcon, and Claudius Masuch. Evolutionary Environment Service Implementation. Deliverable D9.4 Digital Business Ecosystems Project, April 2006.

[4] Claudius Masuch. Evolutionary Environment Network. Webpage. `http://evenet.sourceforge.net`. Last accessed on 06/22/2006.

[5] University of Surrey MUSIC DSG Soluta.net Intel TSSG, UCE. DBE Studio. Webpage. `http://dbestudio.sourceforge.net`. Last accessed on 07/18/2006.

[6] DSG WIT Sun Microsystems, MUSIC. Servent. Webpage. `http://swallow.sourceforge.net/`. Last accessed on 07/18/2006.

[7] Gerard Briscoe and Philippe De Wilde. High-Level Design Specification of the Distributed Intelligence System. Deliverable D6.6 Digital Business Ecosystems Project, December 2005.

[8] DBE Consortium. Addendum No1 - Detailed Work-Plan for the second phase. Sixth Framework Programme, Priority 2.3.1.9., Networked Business and Governments,Proposal/Contract No. IST2002-507953, 2003.

[9] Thomas Kurz, Markus Duerr, Christian Ehgartner, and Roman Greil. Evolutionary Environment Simulator - evesim. Webpage. `http://sourceforge.net/projects/evesim`. Last accessed on 06/22/2006.

[10] Herbert David. *Adaptive Learning by Genetic Algorithms, Second, Revised and Enlarged Edition*. Springer, Berlin Heidelberg New York, 1999.

[11] Swarm Development Group. Java enterprise simulator project - jes. Webpage. `http://www.swarm.org`. Last accessed on 06/22/2006.

[12] Pietro Terna. Java enterprise simulator project - jes. Webpage. `http://web.econ.unito.it/terna/jes/`. Last accessed on 06/22/2006.

[13] Nick Collier, Tom Howe, Robert Najlis, Michael North, and Jerry R. Vos. Recursive porus agent simulation toolkit - repast. Webpage. `http://repast.sourceforge.net`. Last accessed on 06/22/2006.

[14] R. Tobias and C Hofmann. Evaluation of Free Java-libraries for Social-scientific Agent Based Simulation. *Journal of Artificial Societies and Social Simulation*, 7(1), January 2003.

[15] Nick Collier. Properties of nonuniform random graph models. Technical report, Social Science Research Computing, University of Chicago. `http://www.econ.iastate.edu/tesfatsi/RepastTutorial.Collier.pdf`. Last accessed on 06/22/2006.

[16] Heinz Muehlenbein and Thilo Mahnig. Evolutionary algorithms and the boltzmann distribution. In *Foundations of Genetic Algorithms (FOGA2002)*, 2002.

[17] Heinz Muehlenbein and Thilo Mahnig. Evolutionary computation and wright's equation. *Theoretical Computer Science*, September 2002.

[18] Censis. Territorial Social Capital and Driver SMEs. Deliverable D27.1 Digital Business Ecosystems Project, December 2005.

[19] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. The MIT Press, Cambridge, Massachusetts, US, second edition, 2001.

[20] E.W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, pages 269–271, 1959.

[21] M. Petrou, K.N. Giannoutakis, and S. Gautam. Preliminary release of the software of model DBE creation and prediction scenario as a DBE service. Deliverable D12.3 Digital Business Ecosystems Project, 2006.

[22] Mort Bay Consulting. Jetty. Webpage. `http://jetty.mortbay.org/jetty/`. Last accessed on 06/22/2006.

Appendix A

(XSD Validation)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">

<!-- definition of simple elements -->
<xs:element name="smename" type="xs:string"/>
<xs:element name="value" type="xs:decimal"/>
<xs:element name="weight" type="xs:decimal"/>
<xs:element name="useful" type="xs:boolean"/>
<xs:element name="providersme" type="xs:nonNegativeInteger"/>
<xs:element name="trustlevel" type="xs:decimal"/>

<!-- definition of attributes -->
<xs:attribute name="id" type="xs:nonNegativeInteger"/>
<xs:attribute name="pos" type="xs:nonNegativeInteger"/>

<!-- definition of complex elements -->
<xs:element name="partnersme">
  <xs:complexType>
   <xs:all>
    <xs:element ref="trustlevel"/>
   </xs:all>
   <xs:attribute ref="id" use="required"/>
  </xs:complexType>
</xs:element>

<xs:element name="attribute">
  <xs:complexType>
   <xs:all>
    <xs:element ref="value"/>
   </xs:all>
```

```
    <xs:attribute ref="id" use="required"/>
  </xs:complexType>
</xs:element>

<xs:element name="demandedattribute">
  <xs:complexType>
    <xs:all>
      <xs:element ref="value"/>
      <xs:element ref="weight"/>
    </xs:all>
    <xs:attribute ref="id" use="required"/>
  </xs:complexType>
</xs:element>

<xs:element name="attributes">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="attribute" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="demandedattributes">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="demandedattribute" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="historysme">
  <xs:complexType>
    <xs:all>
      <xs:element ref="useful"/>
    </xs:all>
```

```
    <xs:attribute ref="id" use="required"/>
   </xs:complexType>
</xs:element>

<xs:element name="migrationpath">
   <xs:complexType>
    <xs:sequence>
     <xs:element ref="historysme" maxOccurs="unbounded"/>
    </xs:sequence>
   </xs:complexType>
</xs:element>

<xs:element name="serviceonoffer">
   <xs:complexType>
    <xs:all>
     <xs:element ref="attributes"/>
     <xs:element ref="providersme"/>
     <xs:element ref="migrationpath"/>
    </xs:all>
    <xs:attribute ref="id" use="required"/>
    <xs:attribute ref="pos" use="required"/>
   </xs:complexType>
</xs:element>

<xs:element name="servicesonoffer">
   <xs:complexType>
    <xs:sequence>
     <xs:element ref="serviceonoffer" maxOccurs="unbounded"/>
    </xs:sequence>
   </xs:complexType>
</xs:element>

<xs:element name="service">
   <xs:complexType>
    <xs:all>
```

```
      <xs:element ref="attributes"/>
      <xs:element ref="providersme"/>
      <xs:element ref="migrationpath"/>
     </xs:all>
     <xs:attribute ref="id" use="required"/>
   </xs:complexType>
</xs:element>

<xs:element name="localservicepool">
   <xs:complexType>
     <xs:sequence>
       <xs:element ref="service" maxOccurs="unbounded"/>
     </xs:sequence>
   </xs:complexType>
</xs:element>

<xs:element name="demandedservice">
   <xs:complexType>
     <xs:all>
       <xs:element ref="demandedattributes"/>
     </xs:all>
     <xs:attribute ref="id" use="required"/>
   </xs:complexType>
</xs:element>

<xs:element name="usedservice">
   <xs:complexType>
     <xs:all>
       <xs:element ref="attributes"/>
       <xs:element ref="providersme"/>
       <xs:element ref="migrationpath"/>
     </xs:all>
     <xs:attribute ref="id" use="required"/>
   </xs:complexType>
</xs:element>
```

```
<xs:element name="demandedandusedservice">
  <xs:complexType>
   <xs:sequence>
    <xs:element ref="demandedservice" maxOccurs="1"/>
    <xs:element ref="usedservice" minOccurs="0" maxOccurs="1"/>
   </xs:sequence>
   <xs:attribute ref="pos" use="required"/>
  </xs:complexType>
</xs:element>

<xs:element name="demandedandusedservices">
  <xs:complexType>
   <xs:sequence>
    <xs:element ref="demandedandusedservice" maxOccurs="unbounded"/>
   </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="partnersmes">
  <xs:complexType>
   <xs:sequence>
    <xs:element ref="partnersme" maxOccurs="unbounded"/>
   </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="sme">
  <xs:complexType>
   <xs:sequence>
    <xs:element ref="smename"/>
    <xs:element ref="servicesonoffer" minOccurs="0"/>
    <xs:element ref="localservicepool" minOccurs="0"/>
    <xs:element ref="demandedandusedservices" minOccurs="0"/>
    <xs:element ref="partnersmes" minOccurs="0"/>
```

```
    </xs:sequence>
    <xs:attribute ref="id" use="required"/>
  </xs:complexType>
</xs:element>

<xs:element name="model">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="sme" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>
```

Appendix B

(CSV Network Import)

```
;ITA;Barrabes Internet;Gabilos software;Eon;DBS
ITA;x;1;1;1;1
Barrabes Internet;1;x;1;1;1
Gabilos software;1;0;x;0;0
Eon;1;1;1;x;0
DBS;1;0;0;;x
contatto personale;;;;;
```

Appendix C

(Parameter File *Critical Mass Assessment*)

```
<?xml version="1.0" encoding="UTF-8"?>
<parameters>
<model>
<numRuns>50</numRuns>
<stepsToEnd>5000</stepsToEnd>
<numAttributes>20</numAttributes>
<growthFactor>0.0</growthFactor>
<numUselessHops>5</numUselessHops>
<speedFactor>0</speedFactor>
<distanceSME>40</distanceSME>
<mutationFactor>0.0</mutationFactor>
</model>
<smes>
<numSMEs>10</numSMEs>
<numDemandedServicesMin>7</numDemandedServicesMin>
<numDemandedServicesMax>7</numDemandedServicesMax>
</smes>
<services>
<numServices>5</numServices>
<numAttributesMin>15</numAttributesMin>
<numAttributesMax>15</numAttributesMax>
</services>
<localServicePool>
<migrationRate>1.0</migrationRate>
<servicePoolExchange>true</servicePoolExchange>
<servicePoolExchangeFactor>0.2</servicePoolExchangeFactor>
<shrinkByProfile>true</shrinkByProfile>
<shrinkByUsage>false</shrinkByUsage>
<shrinkFactor>0.375</shrinkFactor>
</localServicePool>
```

```
<fileHandling>
<saveModelPath/>
<saveSmeInteractionPath/>
<saveFitnessPath/>
</fileHandling>
<ga>
<populationSize>10</populationSize>
<gaRuns>100</gaRuns>
<optimizeXIndividualsInPopulation>10</optimizeXIndividualsInPopulation>
<exchangePairsOfIndividualsInSubset>2</exchangePairsOfIndividualsInSubset>
</ga>
<comparisons>
<useCentralServicePool>true</useCentralServicePool>
<showComparisonGraph>true</showComparisonGraph>
<showFitnessGraph>false</showFitnessGraph>
<usePartnerEdges>true</usePartnerEdges>
<useProducerEdges>true</useProducerEdges>
<useNetworkEdges>true</useNetworkEdges>
</comparisons>
<equilibrium>
<smeGrouping>false</smeGrouping>
<smeGroups>2</smeGroups>
<numSmesPerGroups>5</numSmesPerGroups>
<smeEqualityFactor>0.8</smeEqualityFactor>
</equilibrium>
</parameters>
```

Appendix D

(Parameter File *Clustering*)

```
<?xml version="1.0" encoding="UTF-8"?>
<parameters>
<model>
<numRuns>1</numRuns>
<stepsToEnd>15000</stepsToEnd>
<numAttributes>20</numAttributes>
<growthFactor>0.0</growthFactor>
<numUselessHops>5</numUselessHops>
<speedFactor>10</speedFactor>
<distanceSME>40</distanceSME>
<mutationFactor>0.0</mutationFactor>
</model>
<smes>
<numSMEs>5</numSMEs>
<numDemandedServicesMin>5</numDemandedServicesMin>
<numDemandedServicesMax>15</numDemandedServicesMax>
</smes>
<services>
<numServices>10</numServices>
<numAttributesMin>2</numAttributesMin>
<numAttributesMax>5</numAttributesMax>
</services>
<localServicePool>
<migrationRate>1.0</migrationRate>
<servicePoolExchange>true</servicePoolExchange>
<servicePoolExchangeFactor>0.1</servicePoolExchangeFactor>
<shrinkByProfile>true</shrinkByProfile>
<shrinkByUsage>false</shrinkByUsage>
<shrinkFactor>0.177</shrinkFactor>
</localServicePool>
<fileHandling>
```

<saveModelPath/>
<saveSmeInteractionPath/>
<saveFitnessPath/>
</fileHandling>
<ga>
<populationSize>20</populationSize>
<gaRuns>1000</gaRuns>
<optimizeXIndividualsInPopulation>10</optimizeXIndividualsInPopulation>
<exchangePairsOfIndividualsInSubset>2</exchangePairsOfIndividualsInSubset>
</ga>
<comparisons>
<useCentralServicePool>false</useCentralServicePool>
<showComparisonGraph>false</showComparisonGraph>
<showFitnessGraph>true</showFitnessGraph>
<usePartnerEdges>true</usePartnerEdges>
<useProducerEdges>true</useProducerEdges>
<useNetworkEdges>true</useNetworkEdges>
</comparisons>
<equilibrium>
<smeGrouping>true</smeGrouping>
<smeGroups>2</smeGroups>
<numSmesPerGroups>3</numSmesPerGroups>
<smeEqualityFactor>0.9</smeEqualityFactor>
</equilibrium>
</parameters>

Appendix E

(Parameter File *Usage Based Clustering*)

```
<?xml version="1.0" encoding="UTF-8"?>
<parameters>
<model>
<numRuns>1</numRuns>
<stepsToEnd>15000</stepsToEnd>
<numAttributes>20</numAttributes>
<growthFactor>0.0</growthFactor>
<numUselessHops>5</numUselessHops>
<speedFactor>10</speedFactor>
<distanceSME>40</distanceSME>
<mutationFactor>0.0</mutationFactor>
</model>
<smes>
<numSMEs>5</numSMEs>
<numDemandedServicesMin>5</numDemandedServicesMin>
<numDemandedServicesMax>15</numDemandedServicesMax>
</smes>
<services>
<numServices>10</numServices>
<numAttributesMin>2</numAttributesMin>
<numAttributesMax>5</numAttributesMax>
</services>
<localServicePool>
<migrationRate>1.0</migrationRate>
<servicePoolExchange>true</servicePoolExchange>
<servicePoolExchangeFactor>0.1</servicePoolExchangeFactor>
<shrinkByProfile>true</shrinkByProfile>
<shrinkByUsage>false</shrinkByUsage>
<shrinkFactor>0.177</shrinkFactor>
</localServicePool>
<fileHandling>
```

<saveModelPath/>
<saveSmeInteractionPath>C:\Dokumente und Einstellungen\Markus\Eigene Dateien\int3.csv</saveSmeInteractionPath>
<saveFitnessPath/>
</fileHandling>
<ga>
<populationSize>20</populationSize>
<gaRuns>1000</gaRuns>
<optimizeXIndividualsInPopulation>10</optimizeXIndividualsInPopulation>
<exchangePairsOfIndividualsInSubset>2</exchangePairsOfIndividualsInSubset>
</ga>
<comparisons>
<useCentralServicePool>true</useCentralServicePool>
<showComparisonGraph>true</showComparisonGraph>
<showFitnessGraph>true</showFitnessGraph>
<usePartnerEdges>true</usePartnerEdges>
<useProducerEdges>true</useProducerEdges>
<useNetworkEdges>true</useNetworkEdges>
</comparisons>
<equilibrium>
<smeGrouping>true</smeGrouping>
<smeGroups>2</smeGroups>
<numSmesPerGroups>3</numSmesPerGroups>
<smeEqualityFactor>0.9</smeEqualityFactor>
</equilibrium>
</parameters>