Workpackage 9
Model of Fitness Landscape

Deliverable 9.3/9.4
Evolutionary Environment Service Implementation

Project funded by the European Community under the “Information Society Technology” Programme
Contract number: 507953
Project acronym: DBE
Title: Digital Business Ecosystem

Deliverable No: D9.3/9.4
Due date: May 31, 2006
Delivery date: May 30, 2006

Short description:
This report contains the technical documentation for the Evolutionary Environment (EvE) implementation.

Author: STU (Thomas J. Heistracher, Thomas Kurz, Giulio Marcon, and Claudius Masuch)
Partners contributed: SUN, INTEL, LSE, ISUI, UBham, HWU, Soluta.net
Made available to: DBE Consortium and European Commission

Versioning

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Author, Organisation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>May 30, 2006</td>
<td>Claudius Masuch (STU)</td>
</tr>
<tr>
<td>0.9</td>
<td>May 7, 2006</td>
<td>Claudius Masuch (STU)</td>
</tr>
<tr>
<td>0.2</td>
<td>April 28, 2006</td>
<td>Claudius Masuch (STU)</td>
</tr>
<tr>
<td>0.1</td>
<td>April 26, 2006</td>
<td>Claudius Masuch (STU)</td>
</tr>
</tbody>
</table>

Quality check
1st internal reviewer: Juanjo Aparicio, SUN
2nd internal reviewer: John M.Kennedy, Intel

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.
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

Table of Contents

1 Introduction 2
   1.1 Document structure 2
   1.2 Glossary 3

2 Design Overview 5
   2.1 Habitat Service internal design 5
      2.1.1 Network Manager 6
      2.1.2 Service Manager 7
      2.1.3 GA and FFF 7
      2.1.4 DIS 7
   2.2 Habitat Interactions 7

3 Source Code Structure 9
   3.1 Build process 9
   3.2 Source Code Structure 10
      3.2.1 eve-common 10
      3.2.2 eve-core 10
      3.2.3 eve-intelligence 11
   3.3 Site Documentation 11

4 Future work 13

5 Package org.dbe.eve 14
   5.1 Interfaces 15
      5.1.1 INTERFACE HabitatService 15
      5.1.2 INTERFACE ServicePool 17
      5.1.3 INTERFACE Solution 17
   5.2 Classes 18
      5.2.1 CLASS EvEService 18
      5.2.2 CLASS HabitatServiceImpl 21
      5.2.3 CLASS SBVRDescription 25
5.2.4 Class ServiceManager ........................................ 26
5.2.5 Class ServicePoolImpl ........................................ 27
5.2.6 Class ServicePoolWorkerTask ................................. 30
5.2.7 Class SolutionSet .............................................. 31

6 Package org.dbe.eve.net ........................................ 34
   6.1 Interfaces ...................................................... 36
       6.1.1 Interface ConnectionPool ................................ 36
       6.1.2 Interface MigrationProbability .......................... 36
       6.1.3 Interface NetworkManager ............................... 37
   6.2 Classes ......................................................... 38
       6.2.1 Class ConnectedNode .................................... 38
       6.2.2 Class ConnectionPoolImpl .............................. 41
       6.2.3 Class DefaultProbability ................................. 43
       6.2.4 Class MigrationWorkerTask .............................. 44
       6.2.5 Class Migrator ........................................... 45
       6.2.6 Class NetworkManagerImpl .............................. 46
       6.2.7 Class NodeID ............................................. 48
       6.2.8 Class SMHandler ......................................... 49

7 Package org.dbe.eve.ga .......................................... 50
   7.1 Interfaces ...................................................... 51
       7.1.1 Interface GeneticAlgorithm ............................... 51
       7.1.2 Interface PrivateSolution ................................ 52
   7.2 Classes ......................................................... 52
       7.2.1 Class GAFactory ......................................... 52
       7.2.2 Class StopCondition ..................................... 53

8 Package org.dbe.eve.ga.impl .................................. 55
   8.1 Classes ......................................................... 56
       8.1.1 Class DefaultGAFactory ................................. 56
       8.1.2 Class DefaultGeneticAlgorithmImpl ................... 57
       8.1.3 Class SimpleBreeder ..................................... 58
       8.1.4 Class SimpleDescription ................................. 59
       8.1.5 Class SimplePopulation .................................. 60
       8.1.6 Class SolutionImpl ....................................... 61
       8.1.7 Class Statistics .......................................... 63
       8.1.8 Class StatsWriter ......................................... 65

9 Package org.dbe.eve.fff ........................................ 66
   9.1 Interfaces ...................................................... 67
       9.1.1 Interface FitnessFunction ............................... 67
       9.1.2 Interface FitnessFunctionAggregator ................... 67
9.2 Classes ................................................................. 68
  9.2.1 CLASS Fitness ............................................ 68
  9.2.2 CLASS FitnessFunctionContext ...................... 69
  9.2.3 CLASS FitnessFunctionFramework ................... 70

10 Package org.dbe.eve.fff.impl 71
  10.1 Classes .......................................................... 72
      10.1.1 CLASS AnotherIFImpl .................................. 72
      10.1.2 CLASS DefaultAggregator ............................. 72
      10.1.3 CLASS FFSimpleImplementation .................... 73

11 Package org.dbe.eve.util 75
  11.1 Classes .......................................................... 76
      11.1.1 CLASS InformationHandler ........................... 76
      11.1.2 CLASS PopulationHelper ............................. 76
      11.1.3 CLASS ServiceTest ................................. 77

List of Abbreviations 80
Chapter 1

Introduction

This report is part of WP9, Model of Fitness Landscape, and relates to the work performed in task C38 - Evolutionary Environment Service Implementation. The report is a technical document describing the EvE service implementation and does not cover general information on the DBE’s evolutionary architecture. In-depth information on the DBE’s evolutionary architecture can be found in deliverables D6.1, D21.1, D9.1 and associated material [1, 2, 3, 4].

As the software described here is work-in-progress, this report is a snap-shot of the status of the EvE habitat service implementation at the delivery date. Up-to-date information on this software’s status can be found on the software project’s source-forges web-site @ http://evenet.sourceforge.net. The source code, which is a part of this deliverable, can be reviewed and downloaded at the project’s subversion repository @ https://svn.sourceforge.net/evenet.

1.1 Document structure

This document is structured into four main parts:

1. EvE Design Overview (Chapter 2)
2. Code Structure Documentation (Chapter 3)
3. Future Work (Chapter 4)
4. JavaDoc\(^1\) generated source-code documentation (Chapter 5 - 11)

The design overview in Chapter 2 shows how the EvE service is integrated within the overall Digital Business Ecosystem (DBE) architecture, and which other DBE components and services are used and called by the EvE. Also, an overview of the logical internal parts of the EvE is provided, together with a brief explanation on

\(^1\)See the JavaDoc tool homepage @ http://java.sun.com/j2se/javadoc/ for additional information
the purpose of each part. The design overview may show functionality that has not be implemented yet. However, this is clearly stated where it applies.

The Code Structure Documentation in Chapter 3 explains how to set up the build process, and describes which libraries are created and used by the distinct parts of the habitat service. In addition, it explains how to modify the project’s documentation web-site.

Chapter 4 gives a brief outlook on the forth-coming work on the EvE implementation.

Finally, the JavaDoc API documentation shows the details of each class and interface provided.

1.2 Glossary

Some terms that are used in this report are well-known within the DBE project community. However, some of them are briefly explained in order to avoid misunderstandings or misinterpretation of concepts.

**EvE Habitat Service**

*also: Habitat or Habitat Service or Node*

The Habitat Service is the software component this reports describes.

**Evolutionary Environment (EvE)**

The Evolutionary Environment (EvE) is the mass of all Habitat Services that are deployed and running in the DBE Service infrastructure

**EvEService**

An EvEService is the representation of a software component or real-world service description within the EvE. An EvEService always points to the Service Manifest that was associated to a service when it has been deployed in the DBE’s Execution Environment. It does not comprise the implementation of the service.

**Migration**

*also used: Service Migration*

Migration is the process of copying or moving an EvEService from one Node’s Service Pool to another.
Service Pool

A Service Pool is a container where all EvE Services of one EvE Habitat Service are stored. A service pool is associated with exactly one habitat and can only be directly accessed by its habitat.

Connection Pool

A Connection Pool is a container where the information on connections to a Habitat’s neighbouring nodes is stored.
Chapter 2
Design Overview

The EvE Habitat Service is implemented as a software service on top of the DBE’s Servent service platform\(^1\). It therefore does not have its own networking protocols or layers, but uses the Peer-to-Peer (P2P) infrastructure provided by the ExE to connect and communicate with other Habitats and with other DBE components. An EvE Habitat Service consists of various parts and components, and plugs into the DBE’s Execution Environment (ExE). Also, the Habitat Service interacts with other Habitat Services, and other DBE Services like the Knowledge Base and the Semantic Registry. The next Sections show and explain the internal design of the Habitat Service as well as the interactions between two Habitat Services, and interaction with other DBE core components.

2.1 Habitat Service internal design

As can be seen in Fig. 2.1, the Habitat Service consists of three main internal modules:

- The *networking* module, which is responsible for building and servicing the habitat network,
- the *service* module that manages all service-pool related tasks,
- the *intelligence* modules, which itself consists of two main components
  - Genetic Algorithm (GA) and Fitness Function Framework (FFF) and
  - Distributed Intelligence System (DIS)

The current release of the EvE Habitat Service does not include a fully complete GA and DIS. However, the design for the implementation of the GA, and the implementation of the Fitness Function Framework (FFF) are provided and implemented in

---
\(^1\)see [3] for more information on the Execution Environment (ExE)
this release. Also, a demo GA and fitness function implementation are provided to give a fast-entry scenario for the final implementation.

![Habitat Design Diagram]

**Figure 2.1: Habitat Design**

The implementation and design of the DIS is not part of WP9 and task C38, and therefore not part of this deliverable.

### 2.1.1 Network Manager

The networking part of the Habitat Service is responsible for all Habitat Network related issues. At a source code level, all related interfaces and classes are placed within the `Package org.dbe.eve.net`. The following activities are handled by the Network Manager:

- Periodically search for new habitats to connect to.
- Manage the Connection Pool, add and remove Nodes, update statistical information about a Node.
- Periodically save the connection pool to the filesystem
- Provide migration probability calculation classes
- Periodically trigger random service migration
- Execute targeted migration (e.g., for the DIS)
- Communication with other DBE components
2.1.2 Service Manager

The service part of the Habitat Service is responsible for all Service Pool related tasks. At a source code level, all related interfaces and classes are placed within the \texttt{-Package org.dbe.eve}. The following activities are handled by the Service Manager:

- Manage the Service Pool, add and remove Services from/to the pool, update statistical information about a Service
- Periodically save the service pool to the filesystem
- Provide "GateKeeper" functionality to the Service Pool.

As the service manager is the single point of access to the Service Pool, checking and testing a service before adding it to the Pool can and should be implemented here.

2.1.3 GA and FFF

As even a brief discussion of the Genetic Algorithms and corresponding Fitness Functions would go beyond the scope of this documentation, we refer the reader to other deliverables that discuss the role of evolutionary computing within the DBE and the EvE, such as D9.1, D8.1 and D6.3 [4, 5, 6]. The current 1.2 implementation also includes the interfaces and classes that allow an easy integration of future GA implementations. At a source code level, these interfaces and classes are bundled within the \texttt{-Package org.dbe.eve.ga}. The software design of the GA interfaces is based on the Factory Design Pattern to allow an easy exchange of implementations without having the necessity to change the actual calling code.

The implementation of the Fitness Function Framework (FFF) that has been presented in [4] can be found within the \texttt{-Package org.dbe.eve.fff}. The current implementation also includes a demo implementation of a FitnessFunction and aggregator. On source code level, this can be found within the \texttt{-Package org.dbe.eve.fff.impl}.

2.1.4 DIS

The Distributed Intelligence System (DIS) is not part of this deliverable. However, as the DIS is a core part of the Evolutionary Environment, the DIS will be a sub-component of the Habitat Service, and therefore accessible only within a Habitat Service.

2.2 Habitat Interactions

Interaction with other DBE components can be divided into interaction with the Service Factory and interaction with the Execution Environment (ExE). The Habitat
Service has to be loosely coupled to other DBE components, meaning that an inability to communicate with a specific DBE component does not lead to a complete failure of the execution of the Habitat Service. However, failures in communication can lead to unforeseen behaviour as data that may be needed for processing might not be available.

Fig. 2.2 gives an overview of the interaction between Habitat Service components, the DBE service factory, and the DBE Execution Environment. Interaction with the Service Factory can be reduced to the interaction with the SBVR Query Tool. The SBVR Query Tool triggers the GA within the Habitat Service, therefore acting as the only end-user triggered functionality. This is done by calling the Habitat Service’s `findBestGuessSolution`-method. As a GA run might take a considerable amount of time, this call is envisioned to be asynchronous. The current implementation is synchronous. As in this case the Habitat Service acts as the callee, failures in the Query Tool should not lead to a failure in the Habitat Service.

Integration with the ExE is mainly needed for two purposes: recognition of a service deployment, and gathering of statistical information about service usage data. This integration is based on the filters functionality of the ExE infrastructure. At the time of writing, the integration has not been implemented.
Chapter 3

Source Code Structure

This chapter describes the structure of the source code and site documentation, and provides information on the libraries being created. Also, the dependencies between the EvE libraries as well as its usage scenarios are clearly outlined.

All code and software that are needed to build the EvE service are provided within the project’s source code repository at http://svn.sourceforge.net/evenet. For detailed information how to access a subversion repository and download the source code see http://subversion.tigris.org.

3.1 Build process

The EvE service is composed of several parts, and has dependencies on other (including 3rd party) software components. To allow a coherent management of all code and dependencies, the EvE uses Maven (http://maven.apache.org) to facilitate the automatic building and distribution of the EvE service and its libraries.

To allow faster development and to avoid version conflicts, Maven Version 1.1 is bundled with the EvE in the /tools/java/maven/maven-1.1 directory.

To start the build process, the following variables must be set:

- JAVA_HOME must point to a valid Java JDK installation
- MAVEN_HOME must point to the provided /tools/java/maven/maven-1.1 directory
- PATH must contain $MAVEN_HOME/bin as an additional entry

Furthermore, the /master/src/config/build.properties file must be copied into the user’s home directory. Script files to automate this set-up process are provided in the bin-directory for *nix (.sh) and windows (.bat) environments.
3.2 Source Code Structure

For a clear and coherent library structure, the source code is divided in three different groups: common, core and intelligence. Each group has one or more sub-projects that usually create a dependent jar file, and may depend upon another. Each group is now introduced briefly.

3.2.1 eve-common

EvE Common provides the generic interfaces and classes of HabitatService. Here, the implementation of an EvEService, a Solution and a SolutionSet, as well as an implementation of the SBVRDescription can be found. Also, the generic Habitat-Service interface is situated in this project.

This library is used by all other EvE libraries, and should be used by any third-party component to communicate with the habitat service. One external user of this library would be the SBVR Query Tool.

<table>
<thead>
<tr>
<th>artifactId:</th>
<th>eve-commons</th>
</tr>
</thead>
<tbody>
<tr>
<td>groupId:</td>
<td>db_ecosystem</td>
</tr>
<tr>
<td>Version:</td>
<td>0.1</td>
</tr>
</tbody>
</table>

3.2.2 eve-core

The EvE core project consists of three sub-projects, filters, habitat-interfaces and habitat-service.

filters

The EvE Filters library will hold the filters functionality for integration with the Execution Environment (ExE). This part is not yet published.

habitat-interfaces

The EvE Core Interfaces Library provides interfaces and classes for the internal communication within the Habitat Service. A user of this library is the DIS.

<table>
<thead>
<tr>
<th>artifactId:</th>
<th>eve-core-interfaces</th>
</tr>
</thead>
<tbody>
<tr>
<td>groupId:</td>
<td>db_ecosystem</td>
</tr>
<tr>
<td>Version:</td>
<td>0.1</td>
</tr>
</tbody>
</table>
habitat-service

This sub-project holds the actual implementation of the Habitat Service. This project will be packed as a deployable archive (DAR) file.

3.2.3 eve-intelligence

The EvE intelligence part contains the interfaces and classes used for the implementation of the Genetic Algorithm (GA), Fitness Function Framework (FFF) and Distributed Intelligence System (DIS).

ga.interfaces

The Genetic Algorithm (GA) interfaces are needed for implementers of a Genetic Algorithm (GA), and for creating and adding new FitnessFunctions to the FFF.

<table>
<thead>
<tr>
<th>artifactId</th>
<th>eve-ga-interfaces</th>
</tr>
</thead>
<tbody>
<tr>
<td>groupId</td>
<td>db_ecosystem</td>
</tr>
<tr>
<td>Version</td>
<td>0.1</td>
</tr>
</tbody>
</table>

ga.core

The GA core provides a demo implementation of a GA with associated FitnessFunctions.

<table>
<thead>
<tr>
<th>artifactId</th>
<th>eve-ga-core</th>
</tr>
</thead>
<tbody>
<tr>
<td>groupId</td>
<td>db_ecosystem</td>
</tr>
<tr>
<td>Version</td>
<td>0.1</td>
</tr>
</tbody>
</table>

To allow for a smooth integration of the eve libraries into projects, the libraries are available at the project’s maven repository @ http://evenet.sourceforge.net/maven/.

3.3 Site Documentation

The EvE Habitat Service was accepted as a sourceforge project (http://sourceforge.net). All source code and related information is available through the project’s sourceforge web-site at http://evenet.sourceforge.net. In order to have all information on one place, the project’s web-site has also been checked in to the project’s source code repository. Following maven’s capability to process the xdoc-format, the whole project web-site has been written using the xdoc format. This documentation can be compiled into html pages by executing the maven site goal.
in the projects site directory. A good starting point for changing and expanding the documentation are the navigation.xml and index.xml file in src/xdocs.
Chapter 4

Future work

As is clear from this report, the EvE Habitat Service implementation is behind its currently envisioned functionality. The next steps in the EvE Habitat Service implementation will focus on the completion of the integration with the ExE, as outlined in Section 2.2, and on fostering the networking and service migration features of the EvE Habitat Network.

As also can be seen, the Habitat Service implementation possibilities strongly depend on the result of the ongoing work and research on other parts of the DBE, especially SBVR. Further development therefore depends on the availability of results in these other parts.
# Chapter 5

## Package org.dbe.eve

### Package Contents

<table>
<thead>
<tr>
<th>Interfaces</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>HabitatService</td>
<td>15</td>
</tr>
<tr>
<td>ServicePool</td>
<td>17</td>
</tr>
<tr>
<td>Solution</td>
<td>17</td>
</tr>
</tbody>
</table>

### Classes

<table>
<thead>
<tr>
<th>Class</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>EvEService</td>
<td>18</td>
</tr>
<tr>
<td>HabitatServiceImpl</td>
<td>21</td>
</tr>
<tr>
<td>SBVRDescription</td>
<td>25</td>
</tr>
<tr>
<td>ServiceManager</td>
<td>26</td>
</tr>
<tr>
<td>ServicePoolImpl</td>
<td>27</td>
</tr>
<tr>
<td>ServicePoolWorkerTask</td>
<td>30</td>
</tr>
<tr>
<td>SolutionSet</td>
<td>31</td>
</tr>
</tbody>
</table>

*This is the main service interface.*

*The EvEService is the individuals within the Digital Ecosystem.*

*This is the CoreAdapter implementation of the HabitatService.*

*The ServiceManager holds the ServicePool, and starts the ServicePoolWorkerTask that (currently) cares about periodically persisting the pool.*

*The ServicePoolImplementation of this HabitatService.*

*The ServicePoolWorkerTask cares about periodic work on the ServicePool.*

*...no description...*
5.1 Interfaces

5.1.1 INTERFACE HabitatService

This is the main service interface. It exposes the methods needed to

- allow habitat-to-habitat communication
- issue a service request
- deploy a new service in this habitat

**Declaration**

```java
public interface HabitatService
```

**Methods**

- `createEvEService`
  ```java
  public void createEvEService(java.lang.String smid,
  org.dbe.eve.SBVRDescription model)
  ```
  - **Usage**
    * Creates a new EvEService with the indicated definition. If the service exists, the old service is overwritten.
  - **Parameters**
    * `smid` - serviceManifest Id of the service, as create by the Semantic Registry
    * `model` - The SBVR Description of the Service

- `findBestGuessSolution`
  ```java
  public SolutionSet findBestGuessSolution(org.dbe.eve.SBVRDescription model)
  ```
  - **Usage**
    * Find the best services that fit the indicated model. This service is only a Guess based on the data the Habitat stores. It usually will not be the real best solution because we do not use ALL the data available in the DBE.
  - **Parameters**
• *model - DBVR model
  – Returns - array with the smids

• *getId
  public String getId()
  – Usage
    * Return the Identifier of this habitat
  – Returns - id The Service Id of this habitat as written in the deployment.properties

• *importEvEService
  public void importEvEService( org.dbe.eve.EvEService service )
  – Usage
    * The send method is invoked by the local HabitatService, if the habitat thinks that a service is worth spreading to the connected habitats

• *recordNewRequest
  public void recordNewRequest( org.dbe.servent.InvokationRequest request )
  – Usage
    * Record a new Request in the Habitat. This request is a call that have been received by the servent.
    TODO: it is possible this InvokationRequest will be changed for a more suitable object
  – Parameters
    * request - invokation request

• *recordNewResponse
  public void recordNewResponse( org.dbe.servent.InvokationResponse response )
  – Usage
    * Record a response for a request in the Habitat.
  – Parameters
    * response - invokation response

• *removeEvEService
  public void removeEvEService( java.lang.String smid )
  – Usage
Remove an EvEService from the pool or form the monitoring services. This will happen when a service will no longer exists in the DBENetwork (usually because it has been undeployed).

- **Parameters**
  - `smid` - serviceManifest identifier (this is the same EvEService identifier)

### 5.1.2 INTERFACE `ServicePool`

#### DECLARATION

```java
public interface ServicePool
    implements java.util.Map, java.io.Serializable
```

#### METHODS

- **`getEvEService`**
  ```java
  public EvEService getEvEService( java.lang.Object i )
  ```
  - **Usage**
    - Same method as Map.get (Object), but specifically to ServicePool, so people that don’t know what this Interface returns can use it.
  - **Parameters**
    - `i` - SM_ID of the Service
  - **Returns** - EvEService returns the associated EvEService

### 5.1.3 INTERFACE `Solution`

#### DECLARATION

```java
public interface Solution
```

D9.3/9.4 Evolutionary Environment Service Implementation Page 17
METHODS

- **getFitnessValue**
  
  ```java
  public float getFitnessValue()
  ```
  
  - **Returns** - float The associated FitnessValue of this Solution

- **getServices**

  ```java
  public EvEService getServices()
  ```
  
  - **Returns** - services The services this solution is composed of

- **getSolutionSize**

  ```java
  public int getSolutionSize()
  ```
  
  - **Returns** - int Number of services the solution is composed of

5.2 Classes

5.2.1 Class EvEService

The EvEService is the individuals within the Digital Ecosystem. They are light-weight entities consisting of a description and a reference to the DBE service they represent.

DECLARATION

```java
public class EvEService
extends java.lang.Object
implements java.io.Serializable
```

SERIALIZABLE FIELDS

- private String homeHabitatId
  -

- private Vector migrationHistory
  -

- private String serviceManifest
• private Vector characteristics
  – This can be removed after the SBVR has been fixed.

• private SBVRDescription sbvr
  –

• private String smId
  – Service manifest identifier

• private String model
  – SBVR model

• private String serviceName
  –

CONSTRUCTORS

• EvEService
  public EvEService( )
  – Usage
    * Empty constructor. Needed for serialization

• EvEService
  public EvEService( java.lang.String smid )
  – Usage
    * Creates a new Object (representation of a real service with this identifier)
  – Parameters
    * smid - service identifier

METHODS

• getCharacteristics
  public Vector getCharacteristics( )
  – Returns - java.util.Vector Returns the characteristics.
• **getHomeHabitatId**
  
  public String getHomeHabitatId( )

  – **Usage**
  
  * The ID of the HomeHabitat of this service. The home habitat is the habitat, where this service has been deployed first.

• **getMigrationHistory**
  
  public Vector getMigrationHistory( )

  – **Usage**
  
  * The Migration History of this service
  
  – **Returns** - A Vector of the id’s of the habitats this service has been

• **getModel**
  
  public String getModel( )

  – **Returns** - Returns the model.

• **getSbvr**
  
  public SBVRDescription getSbvr( )

• **getServiceManifest**
  
  public String getServiceManifest( )

• **getServiceName**
  
  public String getServiceName( )

• **getSmId**
  
  public String getSmId( )

  – **Returns** - Returns the smid.

• **getUsageCounter**
  
  public int getUsageCounter( )

• **setCharacteristics**
  
  public void setCharacteristics( java.util.Vector characteristics )

  – **Parameters**
  
  * characteristics - The characteristics to set.

• **setHomeHabitatId**
  
  public void setHomeHabitatId( java.lang.String homeHabitatId )
Usage
* This should relate to @see org.dbe.eve.HabitatService#getId()

Parameters
* homeHabitatId - The ID of the HomeHabitat

- setMigrationHistory
  public void setMigrationHistory( java.util.Vector migrationHistory )

- setModel
  public void setModel( java.lang.String model )
  Parameters
  * model - The model to set.

- setSbvr
  public void setSbvr( org.dbe.eve.SBVRDescription sbvr )

- setServiceManifest
  public void setServiceManifest( java.lang.String serviceManifest )

- setServiceName
  public void setServiceName( java.lang.String serviceName )

- setSmId
  public void setSmId( java.lang.String smid )
  Parameters
  * smid - The sm_id is the reference to the "real-world" implementation of this service

- setUsageCounter
  public void setUsageCounter( int usageCounter )

5.2.2 Class HabitatServiceImpl

This is the CoreAdapter implementation of the HabitatService
public class HabitatServiceImpl
    extends java.lang.Object
    implements HabitatService, org.dbe.servent.tools.CoreAdapter,
               java.io.Serializable

Serializable Fields

- private String _homeDir
- private String _serviceID
- private String _serventID
- private Vector meteringTransactions
- private Logger _logger
- private String gaName
- private ServiceContext serviceContext
- private ServentContext serverContext
- private long gaMaxIterations
- private float gaMinFitness
- private long gaMaxSeconds
private NetworkManagerImpl networkManager

private ServiceManager serviceManager

CONSTRUCTORS

• HabitatServiceImpl
  public HabitatServiceImpl()
  
  – Usage
    * Empty constructor. It is needed to get no parameter.

METHODS

• createEvEService
  public void createEvEService(java.lang.String smid, org.dbe.eve.SBVRDescription model)
  
  – See Also
    * org.dbe.eve.HabitatService.createEvEService(java.lang.String, org.dbe.eve.SBVRDescription)

• deploy
  public void deploy()

• destroy
  public void destroy()
  
  – See Also
    * org.dbe.servent.Adapter.destroy()

• findBestGuessSolution
  public SolutionSet findBestGuessSolution(org.dbe.eve.SBVRDescription model)
  
  – See Also
    * org.dbe.eve.HabitatService.findBestGuessSolution( org.dbe.eve.SBVRDescription) (in 5.1.1, page 15)
• `getEvEServicePool`
  public ServicePool `getEvEServicePool()`

  — See Also
  * `org.dbe.eve.HabitatServiceImpl.getEvEServicePool()`

• `getHomeDir`
  public String `getHomeDir()`

• `getId`
  public String `getId()`

• `getNetworkManager`
  public NetworkManagerImpl `getNetworkManager()`

• `getServentContext`
  public ServentContext `getServentContext()`

• `getServiceContext`
  public ServiceContext `getServiceContext()`

• `getServiceManager`
  public ServiceManager `getServiceManager()`

• `importEvEService`
  public void `importEvEService( org.dbe.eve.EvEService service )`

  — See Also
  * `org.dbe.eve.HabitatService.importEvEService(org.dbe.eve.EvEService)`
    (in 5.1.1, page 16)

• `importServiceManifest`
  public void `importServiceManifest( java.lang.String smData )`

• `init`
  public void `init( org.dbe.servent.ServentContext context )`

• `init`
  public void `init( org.dbe.servent.ServiceContext context )`

  — See Also
  * `org.dbe.servent.Adapter.init(org.dbe.servent.ServiceContext)`

• `recordNewRequest`
  public void `recordNewRequest( org.dbe.servent.InvokationRequest request )`
See Also

* org.dbe.eve.HabitatService.recordNewRequest(
  org.dbe.servent.InvokationRequest) (in 5.1.1, page 16)

- recordNewResponse
  public void recordNewResponse( org.dbe.servent.InvokationResponse response )

- recordServiceUsage
  public void recordServiceUsage(
    org.dbe.accounting.metering.usagedata.ServiceUsageType serviceUsageType )

- removeEvEService
  public void removeEvEService( java.lang.String smid )

  See Also

  * org.dbe.eve.HabitatService.removeEvEService(java.lang.String) (in 5.1.1, page 16)

- unDeploy
  public void unDeploy( )

5.2.3 Class SBVRDescription

Declaration

public class SBVRDescription
extends java.lang.Object
implements java.io.Serializable

Serializable Fields

- private File modelFile
Constructors

- `SBVRDescription`
  ```
  public SBVRDescription() 
  ```
- `SBVRDescription`
  ```
  public SBVRDescription(java.io.File modelFile) 
  ```
  
  **Usage**
  ```
  * Parse the SBVR File to get a SBVRModel.
  ```

5.2.4 Class ServiceManager

The ServiceManager holds the ServicePool, and starts the ServicePoolWorkerTask that (currently) cares about periodically persisting the pool. Also, it acts as the gateKeeper to the ServicePool, and updates the service information with usage data. It is located within the HabitatServiceImpl.

Declaration

```java
public class ServiceManager
    extends java.lang.Object
```

Constructors

- `ServiceManager`
  ```
  public ServiceManager(org.dbe.eve.HabitatService service) 
  ```

Methods

- `addService`
  ```
  public void addService(org.dbe.eve.EvEService service) 
  ```
- `createEvEService`
  ```
  public void createEvEService(java.lang.String smid, org.dbe.eve.SBVRDescription model) 
  ```
- `getPoolWriterTimer`
  ```
  public TimerTask getPoolWriterTimer() 
  ```
• `getServicePool`
  ```java
  public ServicePool getServicePool()
  ```

• `init`
  ```java
  public void init()
  ```

  – **Usage**
  * The init Method loads the servicePool from a file (if available) and starts the timer for periodic persisting the servicePool to the file

• `recordServiceUsage`
  ```java
  public void recordServiceUsage(java.lang.String smId)
  ```

• `removeService`
  ```java
  public void removeService(java.lang.String smId)
  ```

### 5.2.5 **CLASS ServicePoolImpl**

The ServicePoolImplementation of this HabitatService.

**Declaration**

```java
public class ServicePoolImpl extends java.util.HashMap implements ServicePool, java.io.Serializable
```

**Constructors**

• `ServicePoolImpl`
  ```java
  public ServicePoolImpl()
  ```

• `ServicePoolImpl`
  ```java
  public ServicePoolImpl(org.dbe.servent.ServiceContext context)
  ```

  – **Parameters**
  * context -
Methods

- getEvEService
  public EvEService getEvEService(java.lang.Object i)
  
  - See Also

- persist
  public void persist(java.io.File file)

- unPersist
  public static final ServicePool unPersist(java.io.File file)

Methods inherited from class java.util.HashMap

- clear
  public void clear()

- clone
  public Object clone()

- containsKey
  public boolean containsKey(java.lang.Object )

- containsValue
  public boolean containsValue(java.lang.Object )

- entrySet
  public Set entrySet()

- get
  public Object get(java.lang.Object )

- isEmpty
  public boolean isEmpty()

- keySet
  public Set keySet()

- put
  public Object put(java.lang.Object , java.lang.Object )

- putAll
  public void putAll(java.util.Map )
- **remove**
  ```java
  public Object remove( java.lang.Object )
  ```

- **size**
  ```java
  public int size( )
  ```

- **values**
  ```java
  public Collection values( )
  ```

**METHODS INHERITED FROM CLASS java.util.AbstractMap**

- **clear**
  ```java
  public void clear( )
  ```

- **clone**
  ```java
  protected Object clone( )
  ```

- **containsKey**
  ```java
  public boolean containsKey( java.lang.Object )
  ```

- **containsValue**
  ```java
  public boolean containsValue( java.lang.Object )
  ```

- **entrySet**
  ```java
  public abstract Set entrySet( )
  ```

- **equals**
  ```java
  public boolean equals( java.lang.Object )
  ```

- **get**
  ```java
  public Object get( java.lang.Object )
  ```

- **hashCode**
  ```java
  public int hashCode( )
  ```

- **isEmpty**
  ```java
  public boolean isEmpty( )
  ```

- **keySet**
  ```java
  public Set keySet( )
  ```

- **put**
  ```java
  public Object put( java.lang.Object , java.lang.Object )
  ```

- **putAll**
  ```java
  public void putAll( java.util.Map )
  ```

- **remove**
  ```java
  public Object remove( java.lang.Object )
  ```
5.2.6 CLASS ServicePoolWorkerTask

The ServicePoolWorkerTask cares about periodic work on the ServicePool. At the moment, this is just regularly persisting the ServicePool to a file. But might be used for whatever periodic tasks on the servicePool.

DECLARATION

```java
public class ServicePoolWorkerTask
    extends java.util.TimerTask
```

CONSTRUCTORS

- `ServicePoolWorkerTask`
  ```java
  public ServicePoolWorkerTask( org.dbe.eve.ServicePool servicePool,
                              java.lang.String fileName )
  ```

METHODS

- `run`
  ```java
  public void run( )
  ```

METHODS INHERITED FROM CLASS java.util.TimerTask

- `cancel`
  ```java
  public boolean cancel( )
  ```
- `run`
  ```java
  public abstract void run( )
  ```
- `scheduledExecutionTime`
  ```java
  public long scheduledExecutionTime( )
  ```
5.2.7 CLASS SolutionSet

DECLARATION

public class SolutionSet
    extends java.util.TreeSet

CONSTRUCTORS

- SolutionSet
  public SolutionSet()  

METHODS

- add
  public boolean add(java.lang.Object arg0 )  
- addAll
  public boolean addAll(java.util.Collection arg0 )  
- clear
  public void clear( )  
- contains
  public boolean contains(java.lang.Object o )  
- containsAll
  public boolean containsAll(java.util.Collection arg0 )  
- isEmpty
  public boolean isEmpty( )  
- iterator
  public Iterator iterator( )  
- remove
  public boolean remove(java.lang.Object o )  
- removeAll
  public boolean removeAll(java.util.Collection arg0 )  
- retainAll
  public boolean retainAll(java.util.Collection arg0 )
• **size**
  
  ```java
  public int size()
  ```

• **toArray**
  
  ```java
  public Object toArray()
  ```

• **toArray**
  
  ```java
  public Object toArray(java.lang.Object[] arg0)
  ```

**METHODS INHERITED FROM CLASS java.util.TreeSet**

• **add**
  
  ```java
  public boolean add(java.lang.Object)
  ```

• **addAll**
  
  ```java
  public boolean addAll(java.util.Collection)
  ```

• **clear**
  
  ```java
  public void clear()
  ```

• **clone**
  
  ```java
  public Object clone()
  ```

• **comparator**
  
  ```java
  public Comparator comparator()
  ```

• **contains**
  
  ```java
  public boolean contains(java.lang.Object)
  ```

• **first**
  
  ```java
  public Object first()
  ```

• **headSet**
  
  ```java
  public SortedSet headSet(java.lang.Object)
  ```

• **isEmpty**
  
  ```java
  public boolean isEmpty()
  ```

• **iterator**
  
  ```java
  public Iterator iterator()
  ```

• **last**
  
  ```java
  public Object last()
  ```

• **remove**
  
  ```java
  public boolean remove(java.lang.Object)
  ```

• **size**
  
  ```java
  public int size()
  ```

• **subSet**
  
  ```java
  public SortedSet subSet(java.lang.Object, java.lang.Object)
  ```

• **tailSet**
  
  ```java
  public SortedSet tailSet(java.lang.Object)
  ```
Methods inherited from class java.util.AbstractSet

- equals
  public boolean equals(java.lang.Object)
- hashCode
  public int hashCode()
- removeAll
  public boolean removeAll(java.util.Collection)

Methods inherited from class java.util.AbstractCollection

- add
  public boolean add(java.lang.Object)
- addAll
  public boolean addAll(java.util.Collection)
- clear
  public void clear()
- contains
  public boolean contains(java.lang.Object)
- containsAll
  public boolean containsAll(java.util.Collection)
- isEmpty
  public boolean isEmpty()
- iterator
  public abstract Iterator iterator()
- remove
  public boolean remove(java.lang.Object)
- removeAll
  public boolean removeAll(java.util.Collection)
- retainAll
  public boolean retainAll(java.util.Collection)
- size
  public abstract int size()
- toArray
  public Object[] toArray()
- toArray
  public Object[] toArray(java.lang.Object[])
- toString
  public String toString()
# Chapter 6

## Package org.dbe.eve.net

<table>
<thead>
<tr>
<th>Package Contents</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Interfaces</strong></td>
<td></td>
</tr>
<tr>
<td>ConnectionPool</td>
<td>36</td>
</tr>
<tr>
<td>The ConnectionPool of a Habitat.</td>
<td></td>
</tr>
<tr>
<td>MigrationProbability</td>
<td>36</td>
</tr>
<tr>
<td>Interface for Implementations of the MigrationProbability.</td>
<td></td>
</tr>
<tr>
<td>NetworkManager</td>
<td>37</td>
</tr>
<tr>
<td>The Networkmanager handles the connections between the nodes of the Habitat Network, and cares about the migration and distribution of services. Also, the Networkmanager acts as the &quot;Gatekeeper&quot; to the service pool.</td>
<td></td>
</tr>
<tr>
<td><strong>Classes</strong></td>
<td></td>
</tr>
<tr>
<td>ConnectedNode</td>
<td>38</td>
</tr>
<tr>
<td>A ConnectedNode is the wrapper for a connected habitat Service.</td>
<td></td>
</tr>
<tr>
<td>ConnectionPoolImpl</td>
<td>41</td>
</tr>
<tr>
<td>Implementation of a ConnectionPool.</td>
<td></td>
</tr>
<tr>
<td>DefaultProbability</td>
<td>43</td>
</tr>
<tr>
<td>DefaultProbability if no better MigrationProbability is implemented or available.</td>
<td></td>
</tr>
<tr>
<td>MigrationWorkerTask</td>
<td>44</td>
</tr>
<tr>
<td>The MigrationWorkerTask periodically starts the random migration of services.</td>
<td></td>
</tr>
<tr>
<td>Migrator</td>
<td>45</td>
</tr>
<tr>
<td>The migrator handles the periodic, automatic migration of services.</td>
<td></td>
</tr>
<tr>
<td>NetworkManagerImpl</td>
<td>46</td>
</tr>
<tr>
<td>The NetworkManager cares about all HabitatNetwork - related operations, like sending/receiving services, allow random and targeted migration, etc.</td>
<td></td>
</tr>
<tr>
<td>NodeID</td>
<td>48</td>
</tr>
</tbody>
</table>
Helper class for creating a unique nodeID for a HabitatService if no specific name is given.

SMHandler

The SMHandler (or ServiceManifestHandler) is the EvE’s bridge to the ExE/Service Factory, as here the serviceManifest is read and associated to an EvE Service.
6.1 Interfaces

6.1.1 INTERFACE ConnectionPool

The ConnectionPool of a Habitat. Holds information about all known Habitats, should be persisted regularly and loaded on start-up.

DECLARATION

```java
public interface ConnectionPool
    implements java.util.Map, java.io.Serializable
```

METHODS

- `getNode`
  ```java
  public Object getNode(java.lang.Object id)
  ```

  - **Usage**
    * Same method as Map.get(Object), but specifically to ConnectionPool, so people that don’t know what this Interface returns can use it.

  - **Parameters**
    * `id` - ServiceId

  - **Returns** - Connection

6.1.2 INTERFACE MigrationProbability

Interface for Implementations of the MigrationProbability. All MigrationProbability classes must implement this interface.

DECLARATION

```java
public interface MigrationProbability
```
Methods

- getProbability
  
  ```java
  public float getProbability()
  ```

### 6.1.3 INTERFACE NetworkManager

The NetworkManager handles the connections between the nodes of the Habitat Network, and cares about the migration and distribution of services. Also, the NetworkManager acts as the "Gatekeeper" to the service pool. A service is only added to a habitat by using the NetworkManager’s receive method.

#### Declaration

```java
import java.lang.String;

public interface NetworkManager {

    public ConnectionPool getConnectionPool();

    public void migrate( org.dbe.eve.EvEService service, java.lang.String targetHabitatId );
}
```

#### Methods

- getConnectionPool
  
  ```java
  public ConnectionPool getConnectionPool()
  ```

  - **Usage**
    * Returns the connectionPool of this habitat
  
  - **Returns** - ConnectionPool

- migrate
  
  ```java
  public void migrate( org.dbe.eve.EvEService service, java.lang.String targetHabitatId )
  ```

  - **Usage**
    * Method for targeted migration. To be called from sub-components like the DIS, etc. The target habitat does not have to be known (see ConnectionPool ). However, after a successful migration the target habitat is added to the ConnectionPool .
  
  - **Parameters**
    * service - The Service to be migrated
    * targetHabitatId - The habitat where it goes to.
public void receive( org.dbe.eve.EvEService service )

- Usage
  * This method is usually called by the org.dbe.eve.HabitatService (in 5.1.1, page 15). Here, a "gatekeeper" function to "test" a service before it is added to the service pool can be envisioned.

- Parameters
  * service - The service to be added to the pool

public void send( org.dbe.eve.EvEService service )

- Usage
  * Triggers the random migration of the provided service to all connected habitats.

- Parameters
  * service - The service to migrate to connect habitats

6.2 Classes

6.2.1 CLASS ConnectedNode

A ConnectedNode is the wrapper for a connected habitat Service. It mainly holds the statistic information for a connect habitat and a link to this connect HabitatService.

DECLARATION

public class ConnectedNode
extends java.lang.Object
implements java.io.Serializable

SERIALIZABLE FIELDS

• private Vector migrationHistory
  -

• private MigrationProbability migrationProbability
Fields

- public static final int STATUS_KNOWN
- public static final int STATUS_CONNECTED
- public static final int STATUS_TRUSTED

Constructors

- ConnectedNode
  public ConnectedNode()
Methods

- `getConnectionDate`
  ```
  public Date getConnectionDate()
  ```

- `getLastCommunication`
  ```
  public Date getLastCommunication()
  ```

- `getMigrationHistory`
  ```
  public Vector getMigrationHistory()
  ```

- `getMigrationProbabilityValue`
  ```
  public double getMigrationProbabilityValue()
  ```

- `getNodeId`
  ```
  public String getNodeId()
  ```

- `getService`
  ```
  public HabitatService getService()
  ```

- `getServicesReceived`
  ```
  public int getServicesReceived()
  ```

- `getServicesSent`
  ```
  public int getServicesSent()
  ```

- `getStatus`
  ```
  public int getStatus()
  ```

- `migrateService`
  ```
  public void migrateService( org.dbe.eve.EvEService service )
  ```

- `setConnectionDate`
  ```
  public void setConnectionDate( java.util.Date connectionDate )
  ```

- `setLastCommunication`
  ```
  public void setLastCommunication( java.util.Date lastCommunication )
  ```

- `setMigrationHistory`
  ```
  public void setMigrationHistory( java.util.Vector migrationHistory )
  ```

- `setMigrationProbability`
  ```
  public void setMigrationProbability( org.dbe.eve.net.MigrationProbability migrationProbability )
  ```
• `setNodeId`
  ```java
  public void setNodeId( java.lang.String nodeId )
  ```

• `setService`
  ```java
  public void setService( org.dbe.eve.HabitatService service )
  ```

• `setServicesReceived`
  ```java
  public void setServicesReceived( int servicesReceived )
  ```

• `setServicesSent`
  ```java
  public void setServicesSent( int servicesSent )
  ```

• `setStatus`
  ```java
  public void setStatus( int status )
  ```

### 6.2.2 CLASS `ConnectionPoolImpl`

Implementation of a ConnectionPool. A Container that holds all connected Habitats.

#### DECLARATION

```java
public class ConnectionPoolImpl
    extends java.util.HashMap
    implements ConnectionPool
```

#### CONSTRUCTORS

- `ConnectionPoolImpl`
  ```java
  public ConnectionPoolImpl( )
  ```

#### METHODS

- `getNode`
  ```java
  public Object getNode( java.lang.Object id )
  ```

- `persist`
  ```java
  public void persist( java.io.File file )
  ```

- `unPersist`
  ```java
  public static final ConnectionPool unPersist( java.io.File file )
  ```
METHODS INHERITED FROM CLASS java.util.HashMap

- clear
  public void clear()

- clone
  public Object clone()

- containsKey
  public boolean containsKey( java.lang.Object )

- containsValue
  public boolean containsValue( java.lang.Object )

- entrySet
  public Set entrySet()

- get
  public Object get( java.lang.Object )

- isEmpty
  public boolean isEmpty()

- keySet
  public Set keySet()

- put
  public Object put( java.lang.Object, java.lang.Object )

- putAll
  public void putAll( java.util.Map )

- remove
  public Object remove( java.lang.Object )

- size
  public int size()

- values
  public Collection values()

METHODS INHERITED FROM CLASS java.util.AbstractMap

- clear
  public void clear()

- clone
  protected Object clone()
6.2.3 Class DefaultProbability

DefaultProbability if no better MigrationProbability is implemented or available.

Declaration

```java
public class DefaultProbability
    extends java.lang.Object
    implements MigrationProbability
```
CONSTRUCTORS

- DefaultProbability
  public DefaultProbability() 

METHODS

- getProbability
  public float getProbability() 

6.2.4 CLASS MigrationWorkerTask

The MigrationWorkerTask periodically starts the random migration of services. It plainly executes the executeRandomMigration() method of the Migrator class on a regular basis. See also Migrator.start().

DECLARATION

```java
public class MigrationWorkerTask
  extends java.util.TimerTask
```

CONSTRUCTORS

- MigrationWorkerTask
  public MigrationWorkerTask( org.dbe.eve.net.Migrator migrator )

METHODS

- run
  public void run()
DBE Project (Contract N° 507953)

Methods inherited from class java.util.TimerTask

- `cancel`
  public boolean cancel()
- `run`
  public abstract void run()
- `scheduledExecutionTime`
  public long scheduledExecutionTime()

6.2.5 Class Migrator

The migrator handles the periodic, automatic migration of services. Service migration is run periodically, dependent on the `migration.random.timer` property of the service. Starts the MigrationWorkerTask

Declaration

```java
public class Migrator extends java.lang.Object
```

Constructors

- `Migrator`
  public Migrator()

Methods

- `getConnectionPool`
  public ConnectionPool getConnectionPool()
- `getMigrationProbability`
  public MigrationProbability getMigrationProbability()
- `getMigrationTimer`
  public int getMigrationTimer()
- `getServicePool`
  public ServicePool getServicePool()
• `setConnectionPool`
  public void setConnectionPool( org.dbe.eve.net.ConnectionPool connectionPool )

• `setMigrationProbability`
  public void setMigrationProbability( org.dbe.eve.net.MigrationProbability migrationProbability )

• `setMigrationTimer`
  public void setMigrationTimer( int migrationTimer )

• `setServicePool`
  public void setServicePool( org.dbe.eve.ServicePool servicePool )

• `start`
  public void start( )

### 6.2.6 CLASS NetworkManagerImpl

The NetworkManager cares about all HabitatNetwork-related operations, like sending/receiving services, allow random and targeted migration, etc. It also assigns the MigrationProbability-class to the ConnectedNodes.

**DECLARATION**

```java
public class NetworkManagerImpl
  extends java.lang.Object
  implements NetworkManager
```

**FIELDS**

• public Logger logger

**CONSTRUCTORS**

• `NetworkManagerImpl`
  public NetworkManagerImpl( org.dbe.eve.HabitatService habitat )
**Methods**

- `configureNetwork`  
  ```java
  public void configureNetwork(java.lang.String homeDir)
  ```

- `connect`  
  ```java
  public HabitatService connect(java.lang.String nodeId)
  ```

- `findNewHabitats`  
  ```java
  public void findNewHabitats()
  ```

- `getConnectionPool`  
  ```java
  public ConnectionPool getConnectionPool()
  ```

- `getImportedPool`  
  ```java
  public ServicePool getImportedPool()
  ```

- `getNumberOfConnections`  
  ```java
  public int getNumberOfConnections()
  ```

- `getSMHandler`  
  ```java
  public SMHandler getSMHandler()
  ```

- `importEvEServicePool`  
  ```java
  public void importEvEServicePool(java.lang.String smid)
  ```

  **Usage**  
  * This method is to import the while ServicePool of a given Habitat-Service

  **Parameters**  
  * smid - ID of the habitat to import the pool from

- `importServicesFromSR`  
  ```java
  public ServicePool importServicesFromSR(java.lang.String srAdress)
  ```

- `init`  
  ```java
  public void init()
  ```

  **Usage**  
  * NetworkManager initialisation. Load / Create ConnectionPool, load MigrationProbability class etc.

- `migrate`  
  ```java
  public void migrate(org.dbe.eve.EvEService service, java.lang.String targetHabitatId)
  ```
6.2.7 Class NodeID

Helper class for creating a unique nodeId for a HabitatService if no specific name is given.
Code for getshaID and hexEncode taken from org.dbe.kb.p2p.peermanager

DECLARATION

```
public class NodeID extends java.lang.Object
```

CONSTRUCTORS

```
• NodeID
  public NodeID( )
```

METHODS

```
• createNodeID
  public static String createNodeID( )
```
### 6.2.8 Class SMHandler

The SMHandler (or ServiceManifestHandler) is the EvE’s bridge to the ExE/Service Factory, as here the serviceManifest is read and associated to an EvE Service.

#### DECLARATION

```java
public class SMHandler
extends java.lang.Object
```

#### CONSTRUCTORS

- **SMHandler**
  ```java
  public SMHandler(java.net.URL serverURL)
  ```

#### METHODS

- **createSMFromData**
  ```java
  public SM createSMFromData(java.lang.String smData)
  ```

- **getKB**
  ```java
  public KBI getKB()
  ```

- **getSR**
  ```java
  public SRI getSR()
  ```

- **init**
  ```java
  public void init()
  ```

- **listServiceManifests**
  ```java
  public Collection listServiceManifests()
  ```
Chapter 7

Package org.dbe.eve.ga

Package Contents

<table>
<thead>
<tr>
<th>Interfaces</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>GeneticAlgorithm</td>
<td>51</td>
</tr>
<tr>
<td>PrivateSolution</td>
<td>52</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Classes</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>GAFactory</td>
<td>52</td>
</tr>
<tr>
<td>StopCondition</td>
<td>53</td>
</tr>
</tbody>
</table>

This is the GeneticAlgorithm interface.

...no description...

This class is used by the Habitat Service to get an specific GAFactory implementation, this GAFactory implementation will be the only one who knows how to get a new GeneticAlgorithm (some parameters could be necessary in when the constructor is invoqued)

The StopCondition class try to store the three possible conditions that can be specified by the user to stop the GeneticAlgorithm iterations.
7.1 Interfaces

7.1.1 INTERFACE GeneticAlgorithm

This is the GeneticAlgorithm interface. All Genetic Algorithms must implement this interface. The constructor doesn’t matter, because it is created by an interface, usually also provided by the programmer of the algorithm.

Take into account that the Habitat service deployed in the servent will only call this two methods:

- **start** will be called possibly only once, to store the fitnessFunction and the stop condition into member variables

- **getSolutions** will be called more often, each time somebody wants to know the current solution set. Notice that this can be implemented as a thread that executes the `FitnessFunction.evaluate()` methods in background. While executing, clients can access at the Best solution stored at the moment, but possibly not the best one.

**Declaration**

```
public interface GeneticAlgorithm
```

**Methods**

- **getSolutions**
  ```
  public SolutionSet getSolutions()
  ```
  - **Usage**
    - * Return the solutions stored at the moment. Notice that the GeneticAlgorithm can be still running an probably best solutions can be found later
  - **Returns** - the solution set

- **start**
  ```
  public void start(org.dbe.eve.fff.FitnessFunction function,
  org.dbe.eve.ga.StopCondition condition, org.dbe.eve.ServicePool pool)
  ```
  - **Usage**
Start the Genetic Algorithm using the indicated Fitness function. The evaluation will finish when one of the StopCondition will be accomplished.

TODO: At the moment is not necessary, but in the future this "start" method must start or restart the evaluation of a new Thread, so the evaluation can run in background

- **Parameters**
  - *function* - fitnessFuncion
  - *condition* - stop condition
  - *pool* - ServicePool to be used for this GA run

### 7.1.2 Interface PrivateSolution

**Declaration**

```java
public interface PrivateSolution
    implements org.dbe.eve.Solution
```

**Methods**

- `setFitnessValue`
  ```java
  public void setFitnessValue( float fitnessValue )
  ```

### 7.2 Classes

#### 7.2.1 Class GAFactory

This class is used by the Habitat Service to get an specific GAFactory implementation, this GAFactory implementation will be the only one who knows how to get a new GeneticAlgorithm (some parameters could be necessary in when the constructor is invoqued)

**Declaration**

```java
public abstract class GAFactory
    extends java.lang.Object
```
CONSTRUCTORS

- **GAFactory**
  
  ```java
  public GAFactory()
  ```

METHODS

- **getGeneticAlgorithm**
  
  ```java
  public abstract GeneticAlgorithm getGeneticAlgorithm()
  ```

- **getInstance**
  
  ```java
  public static final GAFactory getInstance(java.lang.String name)
  ```

  - **Parameters**
    * `name` - ClassName of Factory
  
  - **Returns**
    - Instance of the Factory if found
  
  - **Exceptions**
    * `java.lang.InstantiationException` - Is thrown if it is not possible to do the Job

7.2.2 CLASS StopCondition

The StopCondition class try to store the three possible conditions that can be specified by the user to stop the GeneticAlgorithm iterations.

Notice that this object only stores values and is the REAL GeneticAlgorithm implementation that need to check this values (each iteration of gap of time) in order to stop the execution. This class will not stop the execution magically.

DECLARATION

```java
public class StopCondition
    extends java.lang.Object
```

CONSTRUCTORS

- **StopCondition**
  
  ```java
  public StopCondition(long maxSeconds, long maxIterations, float minFitness)
  ```
– **Usage**

* Create a new StopCondition. It is necessary to specify: - the maximum seconds we want to wait - the maximum iteration we want to wait - the maximum fitness we want to accomplish

If one of this conditions doesn’t matter or we are too lazy to specify, we can agree that the StopCondition.INDIFFERENT value must be used

**METHODS**

- *getFitPercentage*
  ```java
  public float getFitPercentage()
  ```
  – **Returns** - Returns the fitPercentage.

- *getIterations*
  ```java
  public long getIterations()
  ```
  – **Returns** - Returns the iterations.

- *getSeconds*
  ```java
  public long getSeconds()
  ```
  – **Returns** - Returns the seconds.
Chapter 8

Package org.dbe.eve.ga.impl

Package Contents

<table>
<thead>
<tr>
<th>Classes</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>DefaultGAFactory</td>
<td>56</td>
</tr>
<tr>
<td>Default GAImplementation</td>
<td></td>
</tr>
<tr>
<td>This implementation should only be used to test that all work fine and, maybe, to implement a first &quot;silly&quot; solution.</td>
<td></td>
</tr>
<tr>
<td>DefaultGeneticAlgorithmImpl</td>
<td>57</td>
</tr>
<tr>
<td>Default Generic Algorithm Implementation.</td>
<td></td>
</tr>
<tr>
<td>SimpleBreeder</td>
<td>58</td>
</tr>
<tr>
<td>...no description...</td>
<td></td>
</tr>
<tr>
<td>SimpleDescription</td>
<td>59</td>
</tr>
<tr>
<td>...no description...</td>
<td></td>
</tr>
<tr>
<td>SimplePopulation</td>
<td>60</td>
</tr>
<tr>
<td>...no description...</td>
<td></td>
</tr>
<tr>
<td>SolutionImpl</td>
<td>61</td>
</tr>
<tr>
<td>This class represent one simple solution for a business model.</td>
<td></td>
</tr>
<tr>
<td>Statistics</td>
<td>63</td>
</tr>
<tr>
<td>...no description...</td>
<td></td>
</tr>
<tr>
<td>StatsWriter</td>
<td>65</td>
</tr>
<tr>
<td>...no description...</td>
<td></td>
</tr>
</tbody>
</table>
8.1 Classes

8.1.1 Class DefaultGAFactory

Default GAImplementation
This implementation should only be used to test that all work fine and, maybe, to implement a first "silly" solution.

TODO: After this, and when GOOD Factories has been provided we can destroy this class

DECLARATION

```java
public class DefaultGAFactory
    extends org.dbe.eve.ga.GAFactory
```

CONSTRUCTORS

- `DefaultGAFactory`
  ```java
  public DefaultGAFactory()
  ```

METHODS

- `getGeneticAlgorithm`
  ```java
  public GeneticAlgorithm getGeneticAlgorithm()
  ```
  - See Also
    * `org.dbe.eve.ga.GAFactory.getGeneticAlgorithm()`

METHODS INHERITED FROM CLASS `org.dbe.eve.ga.GAFactory`

( in 7.2.1, page 52)
- `getGeneticAlgorithm`
  ```java
  public abstract GeneticAlgorithm getGeneticAlgorithm()
  ```
- `getInstance`
  ```java
  public static final GAFactory getInstance(java.lang.String name)
  ```
  - Parameters
    * `name` - ClassName of Factory
8.1.2 Class DefaultGeneticAlgorithmImpl

Default Generic Algorithm Implementation. Only to play with it.
This implementation should only be used to test that all work fine and, maybe, to implement a first “silly” solution.
TODO: After this, and when GOOD Factorys has been provided we can destroy this class.

Declaration

```java
public class DefaultGeneticAlgorithmImpl
extends java.lang.Object
implements org.dbe.eve.ga.GeneticAlgorithm
```

Constructors

- `DefaultGeneticAlgorithmImpl`
  ```java
  public DefaultGeneticAlgorithmImpl()
  ```

Methods

- `getSolutions`
  ```java
  public SolutionSet getSolutions()
  ```
  - See Also
    * `org.dbe.eve.GeneticAlgorithm.getSolution()`

- `start`
  ```java
  public void start(
      org.dbe.eve.fff.FitnessFunction function,
      org.dbe.eve.ga.StopCondition condition,
      org.dbe.eve.ServicePool pool
  )
  ```
  - See Also
8.1.3 CLASS SimpleBreeder

DECLARATION

```java
public class SimpleBreeder
    extends java.lang.Object
```

CONSTRUCTORS

- `SimpleBreeder`
  ```java
  public SimpleBreeder()
  ```

- `SimpleBreeder`
  ```java
  public SimpleBreeder( org.dbe.eve.ServicePool pool )
  ```

METHODS

- `breed`
  ```java
  public SimplePopulation breed()
  ```

- `breed`
  ```java
  public SimplePopulation breed( org.dbe.eve.ga.impl.SimplePopulation population )
  ```
  - Parameters
    - * population - that is the breeding source
  - Returns - SimplePopulation The New population, that contains the breded offsprings

- `getPopulation`
  ```java
  public SimplePopulation getPopulation()
  ```
  - Returns - Returns the population.

- `setPopulation`
  ```java
  public void setPopulation( org.dbe.eve.ga.impl.SimplePopulation population )
  ```
  - Parameters
    - * population - The population to set.
8.1.4  **CLASS SimpleDescription**

**DECLARATION**

```java
public class SimpleDescription
    extends org.dbe.eve.SBVRDescription
```

**SERIALIZABLE FIELDS**

- private Vector characteristics

**CONSTRUCTORS**

- `SimpleDescription`
  ```java
  public SimpleDescription()
  ```

- `SimpleDescription`
  ```java
  public SimpleDescription(java.io.File modelFile)
  ```
  - Parameters
    * `modelFile` - The filename of the SBVR Description file

- `SimpleDescription`
  ```java
  public SimpleDescription(java.util.Vector characteristics)
  ```
  - Parameters
    * `characteristics` - The simple characteristics of this description

**METHODS**

- `getCharacteristics`
  ```java
  public Vector getCharacteristics()
  ```
  - Returns - `characteristics` Returns the characteristics.

- `setCharacteristics`
  ```java
  public void setCharacteristics(java.util.Vector characteristics)
  ```
– Parameters
  * characteristics - The characteristics to set.

Methods inherited from class org.dbe.eve.SBVRDescription

( in 5.2.3, page 25)

8.1.5 CLASS SimplePopulation

Declaration

```java
public class SimplePopulation
extends java.lang.Object
implements java.io.Serializable
```

Serializable Fields

- private Vector individuals

Constructors

- SimplePopulation
  public SimplePopulation( )
- SimplePopulation
  public SimplePopulation( int size )
- SimplePopulation
  public SimplePopulation( org.dbe.eve.ga.impl.SimplePopulation population )

  – Usage
    * Creates a new SimplePopulation based on the other population
  – Parameters
    * population -
Methods

- **addIndividual**
  
  ```java
  public void addIndividual( org.dbe.eve.Solution solution )
  ```

- **changeIndividual**
  
  ```java
  public void changeIndividual( int id, org.dbe.eve.Solution solution )
  ```

- **getIndividual**
  
  ```java
  public Solution getIndividual( int id )
  ```

  - **Returns** - Returns the individuals.

- **getIndividuals**
  
  ```java
  public Vector getIndividuals( )
  ```

  - **Returns** - Returns the individuals.

8.1.6 Class SolutionImpl

This class represents one simple solution for a business model. This is a set of services that can fit the model.

Declaration

```java
public class SolutionImpl
    extends java.lang.Object
    implements org.dbe.eve.ga.PrivateSolution, java.io.Serializable
```

Serializable Fields

- **private float fitnessValue**
  
  - FitnessValue

- **private boolean evaluated**
  
  -

- **private EvEService solution**
  
  - Solution enclosed
CONSTRUCTORS

- *SolutionImpl*
  ```java
public SolutionImpl()
```
- *SolutionImpl*
  ```java
  public SolutionImpl(float fitnessValue, org.dbe.eve.EvEService[] solution)
  
  - **Usage**
    * Constructor. If solution = null it will be discarded
  - **Parameters**
    * fitnessValue - Associated Fitness
    * solution - The Set of Services of which this solution consists
  ```

METHODS

- *getFitnessValue*
  ```java
  public float getFitnessValue()
  
  - **Returns** - Returns the fitnessValue.
  ```
- *getServices*
  ```java
  public EvEService getServices()
  
  - **Usage**
    * The Set of Services of which this solution consists
  - **Returns** - EvEService Array of Services
  ```
- *getSolution*
  ```java
  public EvEService getSolution()
  
  - **Returns** - Returns the solution.
  ```
- *getSolutionSize*
  ```java
  public int getSolutionSize()
  
  - **Usage**
    * Return the number of services (0 if none)
  - **Returns** - int the number of services
  ```
- *isEvaluated*
  ```java
  public boolean isEvaluated()
  ```
- **Returns** - Returns the evaluated.

  - **setEvaluated**
    ```java
    public void setEvaluated(boolean evaluated)
    ```

    - **Parameters**
      - `evaluated` - The evaluated to set.

  - **setFitnessValue**
    ```java
    public void setFitnessValue(float fitnessValue)
    ```

  - **setSolution**
    ```java
    public void setSolution(org.dbe.eve.EvEService[] solution)
    ```

    - **Parameters**
      - `solution` - The solution to set.

### 8.1.7 Class Statistics

#### Declaration

```java
public class Statistics
    extends java.lang.Object
```

#### Constructors

- **Statistics**
  ```java
  public Statistics()
  ```

#### Methods

- **getAvgCharacteristics**
  ```java
  public double getAvgCharacteristics()
  ```

  - **Returns** - Returns the avgCharacteristics.

- **getAvgFitness**
  ```java
  public double getAvgFitness()
  ```

  - **Returns** - Returns the avgFitness.
• `getAvgServices`
  public double `getAvgServices`()

  − **Returns** - Returns the avgServices.

• `getGeneration`
  public int `getGeneration`()

  − **Returns** - Returns the generation.

• `getLengthRatio`
  public double `getLengthRatio`()

  − **Returns** - Returns the lengthRatio.

• `getMaxFitness`
  public double `getMaxFitness`()

  − **Returns** - Returns the maxFitness.

• `setAvgCharacteristics`
  public void `setAvgCharacteristics`(double `avgCharacteristics`)

  − **Parameters**
    * `avgCharacteristics` - The avgCharacteristics to set.

• `setAvgFitness`
  public void `setAvgFitness`(double `avgFitness`)

  − **Parameters**
    * `avgFitness` - The avgFitness to set.

• `setAvgServices`
  public void `setAvgServices`(double `avgServices`)

  − **Parameters**
    * `avgServices` - The avgServices to set.

• `setGeneration`
  public void `setGeneration`(int `generation`)

  − **Parameters**
    * `generation` - The generation to set.
- **setLengthRatio**
  public void setLengthRatio( double  lengthRatio )

  - Parameters
    * lengthRatio - The lengthRatio to set.

- **setMaxFitness**
  public void setMaxFitness( double  maxFitness )

  - Parameters
    * maxFitness - The maxFitness to set.

### 8.1.8 CLASS StatsWriter

#### DECLARATION

```java
public class StatsWriter
extends java.lang.Object
```

#### CONSTRUCTORS

- **StatsWriter**
  public StatsWriter( )

- **StatsWriter**
  public StatsWriter( java.io.File  file )

#### METHODS

- **close**
  public void close( )

- **write**
  public void write( org.dbe.eve.ga.impl.Statistics  stats )
Chapter 9

Package org.dbe.eve.fff

Package Contents

<table>
<thead>
<tr>
<th>Interfaces</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>FitnessFunction</td>
<td>67</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>The FitnessFunction is one of the Core-Parts of a Genetic Algorithm.</td>
<td></td>
</tr>
<tr>
<td>FitnessFunctionAggregator</td>
<td>67</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>...no description...</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Classes</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fitness</td>
<td>68</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>...no description...</td>
<td></td>
</tr>
<tr>
<td>FitnessFunctionContext</td>
<td>69</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>...no description...</td>
<td></td>
</tr>
<tr>
<td>FitnessFunctionFramework</td>
<td>70</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>The FitnessFunctionFramework allows to easily add new FitnessFunctions to the evaluation procedure.</td>
<td></td>
</tr>
</tbody>
</table>
9.1 Interfaces

9.1.1 INTERFACE FitnessFunction

The FitnessFunction is one of the Core-Parts of a Genetic Algorithm. Here, the fitness of a solution created by the GA in relation to the provided request (an SBVRDescription in this very case) can be calculated by implementing the evaluate method.

DECLARATION

public interface FitnessFunction

METHODS

• evaluate
  public Fitness evaluate( org.dbe.eve.Solution solution )
  – Usage
    * Evaluate the fitness function with the indicated solution.
  – Parameters
    * solution - The Solution that should be evaluated
  – Returns - fitness The calculated fitness value

• init
  public void init( org.dbe.eve.SBVRDescription description,
                   org.dbe.eve.fff.FitnessFunctionContext context )
  – Usage
    * Init this fitness function. At this moment, we can get any parameter from the FitnessFunctionContext of store this object in order to get the parameters later, at ”evaluation” time. This will depend if we can perform all evaluations taking into account the ”global system” progress or not.
  – Parameters
    * description - The request that acts as the evaluation counterpart
    * context - The FitnessFunctionContext

9.1.2 INTERFACE FitnessFunctionAggregator
public interface FitnessFunctionAggregator
    implements FitnessFunction

9.2 Classes
9.2.1 Class Fitness

public class Fitness
    extends java.lang.Object
    implements java.lang.Comparable

    public Fitness()

    public int compareTo(java.lang.Object arg0)

    public float getValue()
        Returns - Returns the value.
• **setValue**
  ```
  public void setValue( float value )
  ```

  - **Parameters**
    * value - The value to set.

### 9.2.2 Class FitnessFunctionContext

#### Declaration

```java
public class FitnessFunctionContext
    extends java.lang.Object
```

#### Constructors

- **FitnessFunctionContext**
  ```
  public FitnessFunctionContext( )
  ```

#### Methods

- **getAttribute**
  ```
  public Object getAttribute( java.lang.Object key )
  ```

  - **Usage**
    * Search for an attribute, if not found, return null
  
  - **Parameters**
    * key - identifier of the attribute
  
  - **Returns** - value (null if not found)

- **putAttribute**
  ```
  public void putAttribute( java.lang.Object key, java.lang.Object value )
  ```

  - **Usage**
    * Add a new and generic attribute to the context
  
  - **Parameters**
    * key - the identifier of the attribute
    * value - value we want to store
9.2.3 CLASS FitnessFunctionFramework

The FitnessFunctionFramework allows to easily add new FitnessFunctions to the evaluation procedure.

Customisation of the Framework is done by adding/changing parameters in the fitnessFramework.properties file.

FitnessFunctions are added by adding the class name to the FitnessFunctions parameter this way: FitnessFunctions : my.fitness.function.1 my.fitness.function.2 my.fitness.function.3 and by making sure that the classes can be found on the classpath.

DECLARATION

```
public final class FitnessFunctionFramework
    extends java.lang.Object
```

FIELDS

- public static final String PROPERTY_FILE
  - name of the property file

METHODS

- `getFitnessFunction`
  ```
  public static FitnessFunction getFitnessFunction(
      org.dbe.eve.SBVRDescription description,
      org.dbe.eve.fff.FitnessFunctionContext context )
  ```

- `getFitnessFunction`
  ```
  public FitnessFunction getFitnessFunction( java.lang.String name )
  ```

- `getFitnessFunctionNames`
  ```
  public String getFitnessFunctionNames( )
  ```
Chapter 10

Package org.dbe.eve.fff.impl

Package Contents

<table>
<thead>
<tr>
<th>Classes</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>AnotherFFImpl</td>
<td>72</td>
</tr>
<tr>
<td>DefaultAgregator</td>
<td>72</td>
</tr>
<tr>
<td>FFSimpleImplementation</td>
<td>73</td>
</tr>
</tbody>
</table>

TODO JavaDoc

A "how-to" implementation example for a FitnessFunctionAggregator.

...no description...
10.1 Classes

10.1.1 CLASS AnotherFFImpl

Todo JavaDoc

Declaration

```java
public class AnotherFFImpl
    extends java.lang.Object
    implements org.dbe.eve.fff.FitnessFunction
```

Constructors

- `AnotherFFImpl`
  ```java
  public AnotherFFImpl()
  ```

Methods

- `evaluate`
  ```java
  public Fitness evaluate( org.dbe.eve.Solution solution )
  ```
- `init`
  ```java
  public void init( org.dbe.eve.SBVRDescription description,
                  org.dbe.eve.fff.FitnessFunctionContext context )
  ```

10.1.2 CLASS DefaultAggregator

A “how-to” implementation example for a FitnessFunctionAggregator. In the evaluate method, all FitnessFunctions registered within the Framework are combined to produce one single FitnessValue out of several FitnessFunctions.

In this example, the overall fitness is just the sum of all evaluation results of the single fitness functions. If necessary, different mechanisms of combining distinct results can be envisioned.
DBE Project (Contract N° 507953)

DECLARATION

```java
public class DefaultAggregator
  extends java.lang.Object
  implements org.dbe.eve.fff.FitnessFunctionAggregator
```

CONSTRUCTORS

- `DefaultAggregator`
  ```java
  public DefaultAggregator()
  ```

METHODS

- `evaluate`
  ```java
  public Fitness evaluate( org.dbe.eve.Solution solution )
  ```
- `init`
  ```java
  public void init( org.dbe.eve.SBVRDescription description,
                   org.dbe.eve.fff.FitnessFunctionContext context )
  ```
- `setFramework`
  ```java
  public void setFramework( org.dbe.eve.fff.FitnessFunctionFramework framework )
  ```

10.1.3 CLASS FFSimpleImplementation

DECLARATION

```java
public class FFSimpleImplementation
  extends java.lang.Object
  implements org.dbe.eve.fff.FitnessFunction
```

CONSTRUCTORS

- `FFSimpleImplementation`
  ```java
  public FFSimpleImplementation()
  ```

D9.3/9.4 Evolutionary Environment Service Implementation Page 73
**Methods**

- **evaluate**
  
  ```java
  public Fitness evaluate( org.dbe.eve.Solution solution )
  ```

- **getDescription**
  
  ```java
  public SBVRDescription getDescription() 
  ```
  
  Returns - Returns the description.

- **init**
  
  ```java
  public void init( org.dbe.eve.SBVRDescription description,
                   org.dbe.eve.fff.FitnessFunctionContext context )
  ```
Chapter 11

Package org.dbe.eve.util

Package Contents

<table>
<thead>
<tr>
<th>Classes</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>InformationHandler</td>
<td>76</td>
</tr>
<tr>
<td>PopulationHelper</td>
<td>76</td>
</tr>
<tr>
<td>ServiceTest</td>
<td>77</td>
</tr>
</tbody>
</table>

...no description...

...no description...

...no description...
11.1 Classes

11.1.1 Class InformationHandler

**Declaration**

```java
public class InformationHandler
    extends java.lang.Object
    implements org.dbe.servent.http.ServiceHandler
```

**Constructors**

- `InformationHandler`
  
  ```java
  public InformationHandler()
  ```

**Methods**

- `handle`
  
  ```java
  public void handle( org.dbe.servent.http.ServentRequest request,
                     org.dbe.servent.http.ServentResponse response )
  ```
  
  - See Also
    
    ```java
    * org.dbe.servent.http.ServiceHandler.handle(
      org.dbe.servent.http.ServentRequest,
      org.dbe.servent.http.ServentResponse)
    ```

- `init`
  
  ```java
  public void init( java.lang.Object service, org.dbe.servent.ServiceContext context )
  ```
  
  - See Also
    
    ```java
    * org.dbe.servent.http.ServiceHandler.init(java.lang.Object,
      org.dbe.servent.ServiceContext)
    ```

11.1.2 Class PopulationHelper
public class PopulationHelper
    extends java.lang.Object

Methods

- getInstance
  public static PopulationHelper getInstance()

- getPoolFromFile
  public ServicePool getPoolFromFile(java.lang.String fileName)

  - Usage
    * Create / Read a service pool from a file
  - Parameters
    * fileName - Filename of the file including extension

- getPopulationFromFile
  public SimplePopulation getPopulationFromFile(java.lang.String fileName)

- main
  public static void main(java.lang.String [] args)

  - Parameters
    * args -

11.1.3 Class ServiceTest

Declaration

public class ServiceTest
    extends java.lang.Object
CONSTRUCTORS

- **ServiceTest**
  
  public ServiceTest()

METHODS

- **main**
  
  public static void main( java.lang.String [] args )

  - Parameters
    
    * args -
Bibliography


List of Abbreviations

**DBE**  Digital Business Ecosystem

**DIS**  Distributed Intelligence System

**EvE**  Evolutionary Environment

**ExE**  Execution Environment

**FFF**  Fitness Function Framework

**GA**  Genetic Algorithm

**HWU**  Heriot-Watt University Edinburgh

**INTEL**  Intel Ireland Ltd.

**ISUFI**  Istituto Superiore Universitario di Formazione Interdisciplinare, Lecce

**LSE**  London School of Economics and Political Science

**P2P**  Peer-to-Peer

**SBVR**  Semantics of Business Vocabulary and Business Rules

**SUN**  SUN Microsystems Iberia

**UBham**  University of Birmingham