GreenFire – Intelligent Heating

1. Introduction

GreenFire [1] is an open source project designed to efficiently manage and control the heating system of a house or apartment and thus save energy. GreenFire is able to control heating remotely and provides a simple Java API to do so. Every five minutes GreenFire checks all the sensors and the weather forecast and then decides what to do. It then sets the heating mode, "On", "Off", etc. The intelligent heating control and monitoring system saves between 20% and 50% energy, is environment friendly, and at the same time does not decrease the level of comfort in your home. The main goal of the project is highest energy savings and to reduce CO2 consumption. As recognition, it has received the JAX Innovation Award.

The main characteristics of GreenFire are as follows:

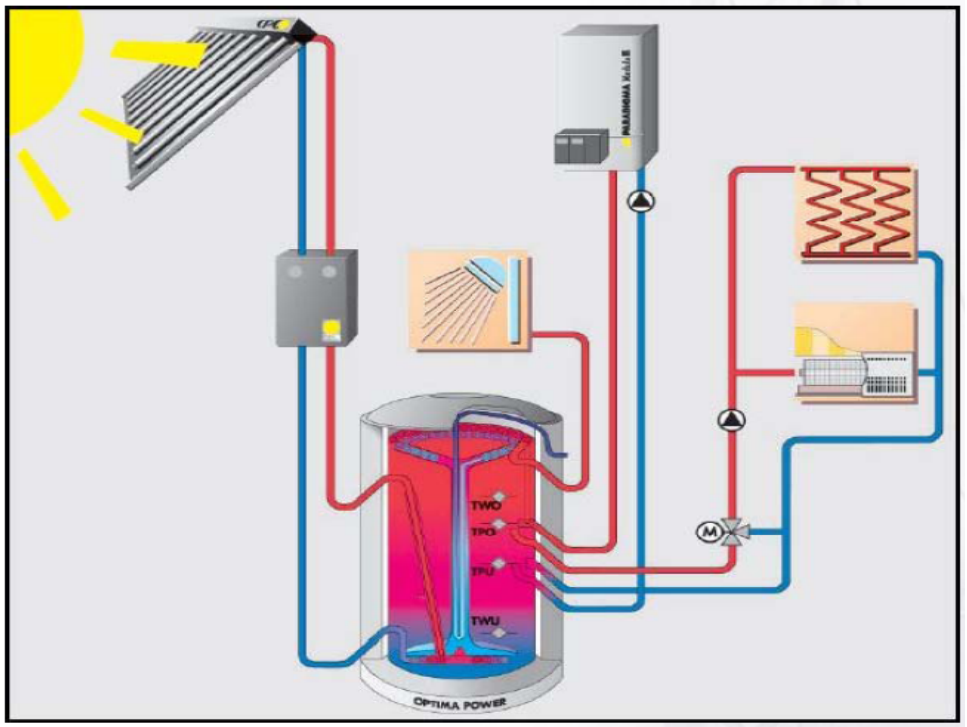
- Management
- Monitoring
- Reporting

- Weather Forecast
- Control Ventilation

Solar heating "*is the usage of solar energy to provide process, space, or water heating*" [2]. Solar energy is cheaper to use than energy based on oil, gas, or wood-pellet, and in addition, it is CO2 neutral. The only downside is that predicting when and for how long the sun will shine, is not an easy task.

In modern heating systems, the solar energy is stored in buffers, which are at the heart of the heating control. The temperature of these buffers affects the decision of whether the primary heating source should be turned on or off. The problem is that today's modern systems use only a time driven approach to control the heating. Several other factors are not taken into account, such as the current solar-power, weather predictions, and internal house temperatures.

Below is an image of a heating system [3]:



Source: paradigma.de

Figure 1 Buffer Storage Tank - the heart of the Heating System

2. Problem Cases

The sun shines, but for example warm water or heating is needed only in a few hours, so the primary energy source (oil/wood pellet) will start to heat the buffer.

Since the temperature increases, the solar collectors are less efficient; hence, they are going to be turned off.
Furthermore, let assume that you are going to make fire in the wood burning stove, which is connected to the buffer. However, your heating system does not know this and uses the primary heating anyway, which causes an energy waist.
In addition, sun may shine, but there is enough energy in the buffer. To avoid overheating, the control decides to turn the solar collectors off, instead of turning on the heating in the basement, which is another way of cooling. In the same situation, and usually if it is warm enough, the house heating, as well as the heater, should be turned off.
If we can predict that the weather for the next day is good, the primary source of heating (oil/wood-pellet) could be earlier turned off, so the buffer could get cooler to utilize the sun at the next day.

3. Conceptual Overview

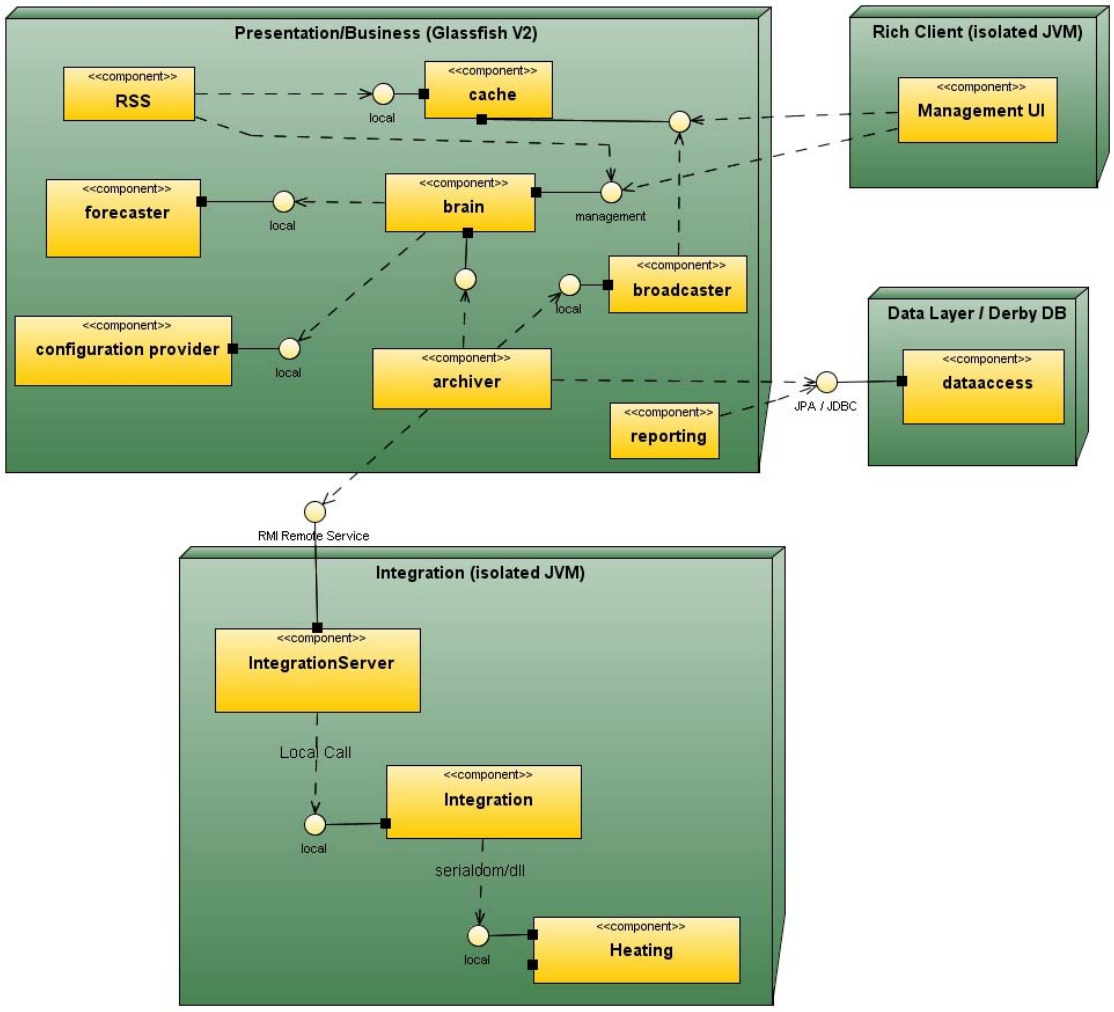


Figure 2 GreenFire's Building Blocks

GreenFire wakes up every 5 minutes (the archiver, actually the heartbeat - implemented by a timer bean) completes the following tasks:

- It gathers the data (heating + configuration, weather forecast) and passes it to the „brain“.
- The brain is asked what to do (heating on/off, etc).
- The decision is passed to the integration layer and executed.
- All the data is stored in the Derby DB and can be used in reports.

The “brain” downloads a script from an URL (Glassfish as well). It has access to all the data (JPA entities) and can decide what to do.
The data in the JavaDB is used for reports and monitoring independently (it stores every 5 minutes all the data since about 2 years – and performs still very well -> no problems). The Reporting-UI is accessing JavaDB to read the decisions and interesting data like external temperature, weather etc. – not directly the heating system (decoupling – the serial port seems not to be thread safe).

4. Technologies Used

To achieve its goals, the following technologies/APIs were used:

- Glassfish V2 [4], an open source application server that implements Java EE 5 [5], and Java SE 6 [6].
- Glassfish Shoal [7], a java-based scalable dynamic clustering framework that provides infrastructure to build fault tolerance, reliability, and availability, is used to publish data in LAN environment to RIAs (Java FX tools etc) – basically to publish and distribute the information over the local network in real time.
- Glassfish advanced monitoring capabilities (AMX, JMX and Node Agents) are used to monitoring the system.
- JSR-223 (Scripting Integration) in Java EE 6 environment for the implementation of flexible rule systems.
- Reporting.
- EJB 3 timer service, which enable you to schedule timed notifications.
- Java EE compatible hardware integration.
- Sun SPOT [8], a wireless sensor network mote developed by Sun Microsystems, and sensor network integration.
- Java FX [9], a family of products for creating rich internet applications (RIA), together with Swing and EJB 3 [10].
- Speech synthesizer integration (FreeTTS) [11], used by GreenFire to read the current state and forecast, and Sphinx [12], a speech recognizer used as a replacement for conventional UI.
- Management and monitoring over the internet (RSS-Feeds, WebClient).

- Mobile device integration, using Java ME [13].
- Integration of Multi Media Center Systems (monitoring and viewers)

To communicate with the heating system, the COM interface is used. For the SunSPOTs, communication is done via air (wirelessly).

For the implementation, the author has used several patterns available in [14]. Unfortunately, to get any insight from these patterns you would need to buy the authors book.

5. Monitoring and Reports

For monitoring the status of a home, one can use RSS feeds. One example of such a feed is presented below:

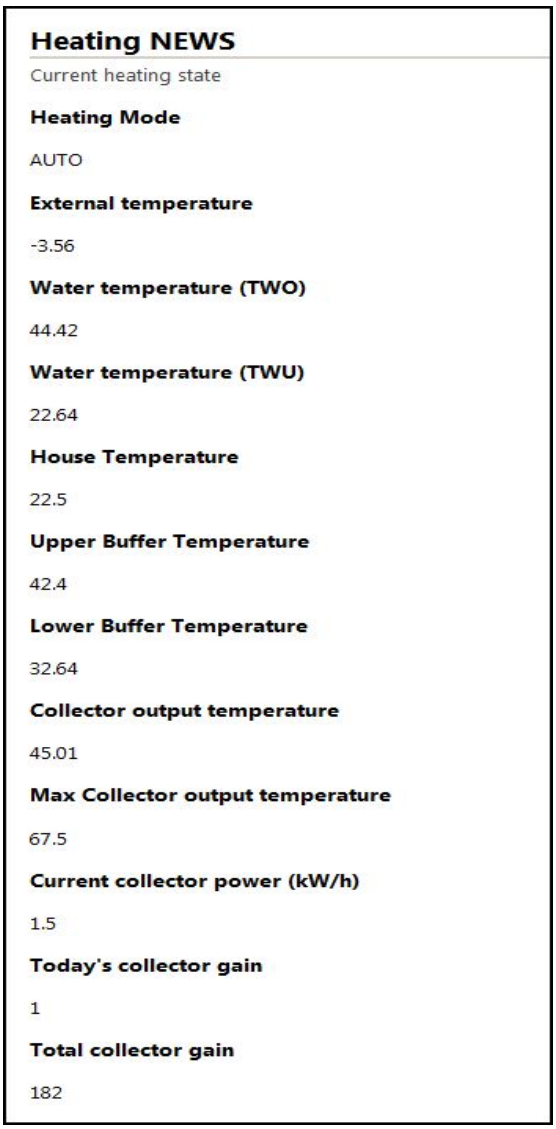


Figure 3 Mobile Report

You can use your own mobile phone to display the information mentioned above.

All decisions and data are available in form of reports. Below is an example of such a report:

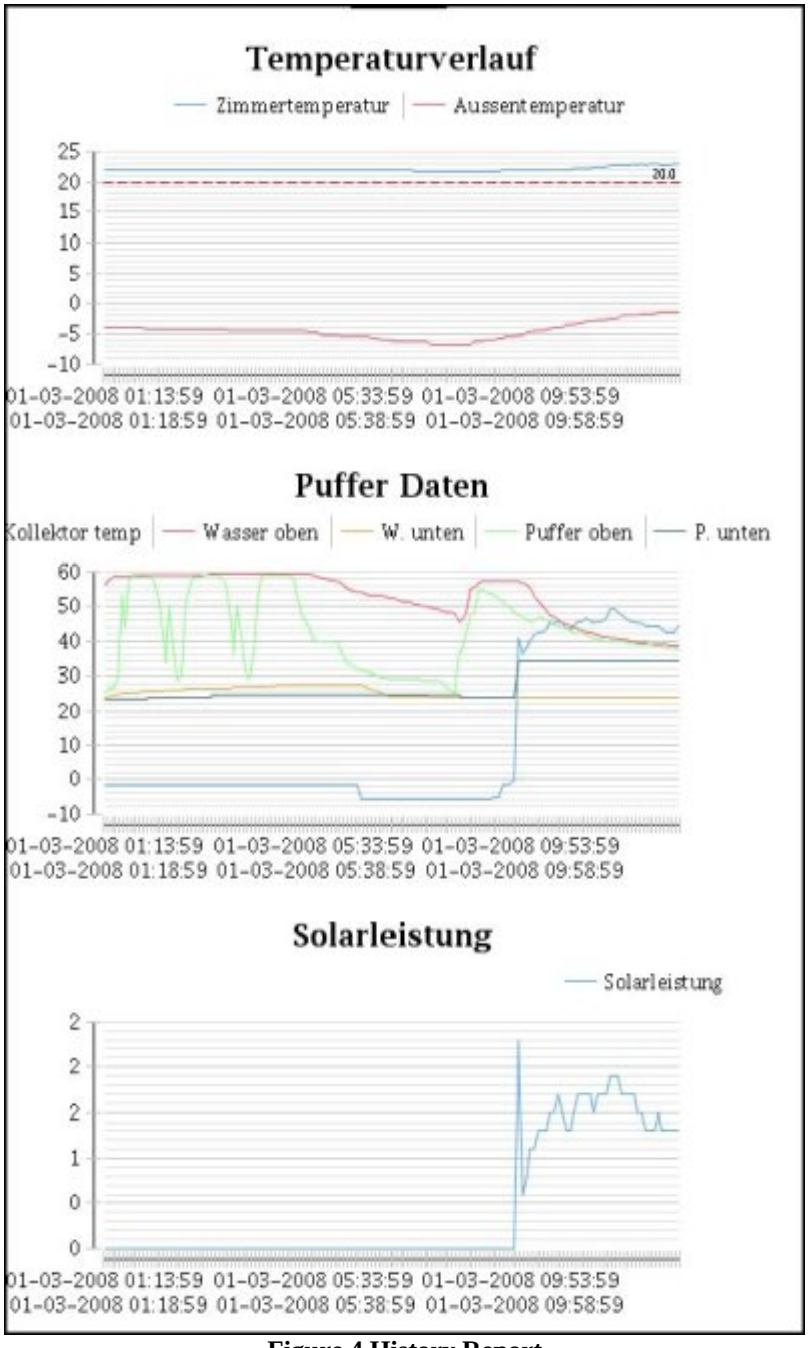


Figure 4 History Report

6. The Gaia Methodology

The question that arises is why we need to describe the heating system in terms of a multiagent system. Firstly, a heating system such as GreenFire is inherently concurrent and distributed. Secondly, GreenFire is an always-on system. Thirdly, there can always be components that join or leave the system, such as wireless sensors that measure the temperature in different rooms. Given all this, designing and developing such a heating system can be done in terms of agents, or autonomous software entities, that can achieve their objectives (sensing, gathering information, processing information, distribute information, store information) by interacting with each other and with the environment. Agents are just software components that have a specific role(s) they need to fulfill. Such roles in GreenFire are the ones mentioned above (sensing, processing, etc).

The hierarchical organization of GreenFire is showed in the figure below:

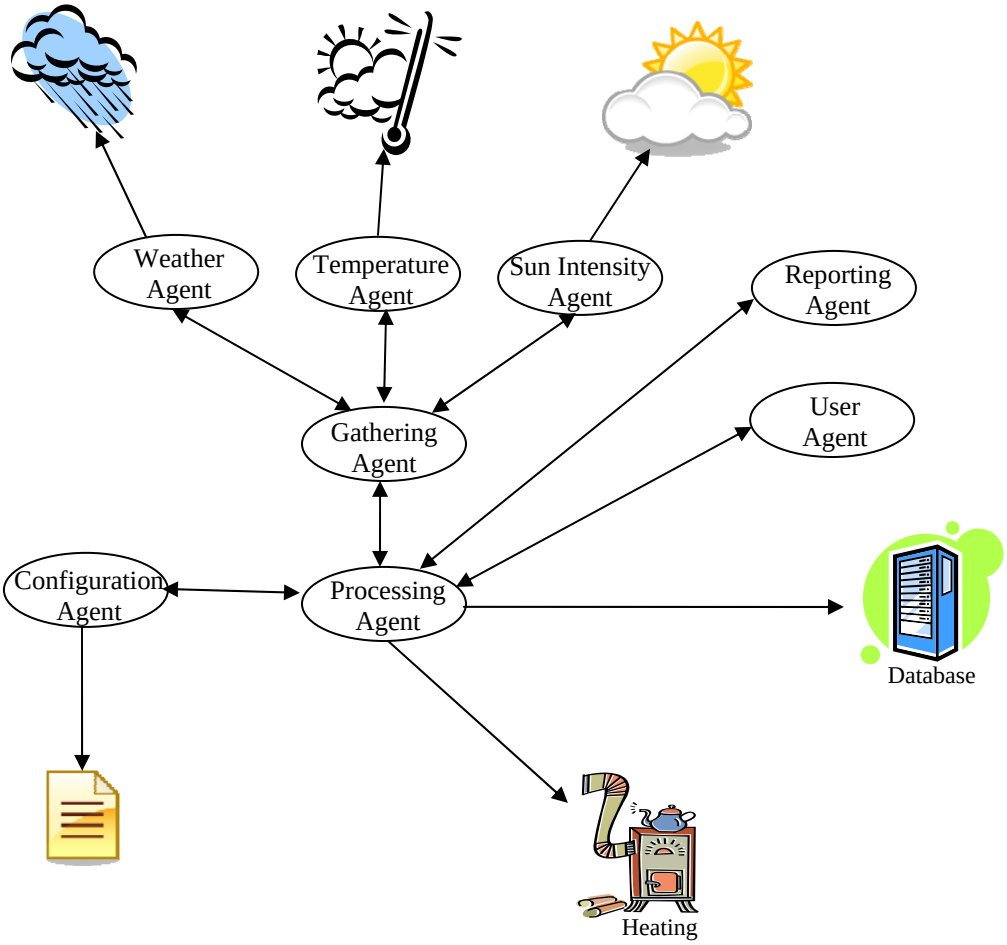


Figure 5 GreenFire's Multi-Level Hierarchical Organization

6.1 The Analysis Phase.

6.1.1. The Organizations

There are several organizations in the heating system. First, the organization responsible for sensing the environment and gathering all the information. Second, the organization responsible for processing and distributing the results. Third, the organization responsible for interacting with the user. For all these case, there is a clear goal to be pursued in each organization. The interactions between the first two organizations are scheduled at specific intervals of time (every five minutes). The interaction between the second and the third organization depends on the user.

6.1.2 The Environmental Model

The idea is to treat the environment in terms of resources made available to agents for sensing and for consuming:

reads	weather, temperature, sun intensity, configuration file
changes	database, heating

6.1.3 The Preliminary Role Model

In this stage, we identify some of the some characteristics (basic skills) of the system that are likely to remain the same independently of the actual organization structure. These skills are called preliminary roles. The basic interactions needs, also called preliminary protocols, are defined.
In the heating system, the preliminary roles are SENSOR, INFORMATIONGATHERER, and DECISIONMAKER. The permissions for each of these roles are specified next. For the Sensor, we have:

reads	Temperature, Sun_Intensity

For InformationGatherer we have:

reads	Sensor_Information

For DecisionMaker, we have:

reads	Configuration_Information, Sensor_Data
changes	Database, Heating

For protocols and activities, we have the following information available for each of the roles defined previously.

SENSOR = (ReadData.ManageData.SendData)

The sequential execution is performed once every five minute. The expression says that the SENSOR consists of executing the protocol ReadData, followed by the activity ManageData and the protocol SendData. manage

INFORMATIONGATHERER = (ReceiveData.HandleData.SendData)

The sequential execution is performed once every five minute. The expression says that the INFORMATIONGATHERER consists of executing the protocol ReceiveData, followed by the activity HandleData and the protocol SendData.

DECISIONMAKER = (GatherData.ProcessData.MakeDecisions.PublishResults)

The expression says that the DECISIONMAKER consists of executing the protocol GatherData, followed by two activities, namely ProcessData and MakeDecission. Lastly, the PublishResults protocol will be executed in parallel because there are several resources that need to be updated, and it can happen at the same time. Let us look at how the schema for role SENSOR looks like:

Role Schema: SENSOR
Description: This preliminary role involves reading sensorial data such as temperature and light, and sending the data further to be managed by a different entity.
Protocols and Activities: ReadData, ManageData, SendData
Permissions: reads Sensor_Information
Responsibilities Liveness: SENSOR = (ReadData.ManageData.SendData)

Below is the schema for the role INFORMATIONGATHERER:

Role Schema: INFORMATIONGATHERER
Description: This preliminary role involves gathering the data from sensors, managing these data and ending it to a decision making entity for further processing and decision making.
Protocols and Activities: ReceiveData, HandleData, SendData
Permissions: reads Sensor_Information
Responsibilities Liveness: INFORMATIONGATHERER = (ReceiveData.HandleData.SendData) Safety: ▪ manage_information = all_sensor_data_receiver.

Below is the schema for the role DECISIONMAKER:

Role Schema: DECISIONMAKER
Description: This preliminary role involves gathering the data from some specialized entity, process the data, make a decision on weather to turn the heating on or off, and publish the results.
Protocols and Activities: GatherData, ProcessData, MakeDecission, PublishResults
Permissions: reads Configuration_Information, Sensor_Data changes Database, Heating
Responsibilities Liveness: DECISIONMAKER = (GatherData.ProcessData.MakeDecision.PublishResults) Safety: ▪ decision_variants = On | Off.

6.1.4 Preliminary Interaction Model

This model captures the dependencies and relationships between the various roles in the multiagent heating system (GreenFire).

The ReadData preliminary protocol definition is shown below:

Protocol name: **Read Data**			
Initiator: **Timer**	Responder: **Sensor**	Partner: None	Input: **timer alert**
Description: Every five minutes, the sensors need to wake up and sense the environment for data related to temperature and light intensity.			Output: **sensor data**

The SendData preliminary protocol definition is shown below:

Protocol name: **Send Data**			
Initiator: **Sensor**	Responder: **Sensor**	Partner: None	Input: **sensor data**
Description: After all the data is managed from the existing sensors, it is grouped into a more processable form and sent for further processing.			Output: **sensor data**

The GatherData preliminary protocol definition si shown below:

Protocol name: **Gather Data**			
Initiator: **Information Gatherer**	Responder: **Decision Maker**	Partner: None	Input: **sensor data**
Description: After the data has been managed by the information gatherer, which data needs to be gathered by the entity that will process it and make decisions based on it.			Output: **processable data**

The PublishResults preliminary protocol definition is shown below:

<table>
<tr><td colspan="3">Protocol name:
Publish Results</td><td></td></tr>
<tr><td>Initiator:
Decision Maker</td><td>Responder:
Decision Maker</td><td>Partner:
None</td><td>Input:
sensor data</td></tr>
<tr><td colspan="3">Description:
After the data has been managed by the information gatherer, which data needs to be gathered by the entity that will process it and make decisions based on it.</td><td>Output:
processable data</td></tr>
</table>

6.1.5 The Organizational Rules

The organizational rules are considered as responsibilities of the organization as a whole. There are two types of rules: safety rules, which define the time-independent global invariants for the organization that must be respected, and liveness rules that define how the dynamics of the organization should evolve over time. Let us look at some of these rules for the heating system GreenFire.
For example, the rule that expresses the fact that managing data cannot happen before all three sensors have sensed the data, is shown below:

MANAGEDATA -> SENSINGDATA

In addition, processing the data cannot happen before the data is actually received from both the configuration files and from the gathering entity:

PROCESSINGDATA -> RECEIVECONFIGDATA, RECEIVESENSDATA

6.2. The Design Phase

The output of the analyzing phase is a documentation of all the functional and some non-functional characteristics of the GreenFire heating system. In addition, the characteristics of the operational environment in which the heating system will be situated are identified. This specification is used in architectural design in a way to structure the multiagent system organization. One of the outcomes could be Figure 1.

From the preliminary roles that we have identified in the analysis phase, we have to decide which becomes an agent and which becomes a service. In the GreenFire heating system, all three roles become agents.

Identifying the organizational abstraction aids in the analysis and design phases of complex systems. After going through some of the stages described above, and through more of the ones described in [15], you get a more insight into the system you want to analyze and design.

7. Future Directions

GreenFire decides about the next actions based on the whether forecast, current collector "power" and the "thermos" temperature. Integrating the internal, external temperature, as well the sun "radiation" in different rooms would be an improvement. One could use SunSPOT for this purpose.
Better reporting could also be achieved. Currently, GreenFire has gathered weather data for about three years in JavaDB. It would be nice to correlate the data, so you would see the temperature, energy costs etc. of the current day, but one or two years ago.
Control the ventilator, which in the author's case runs all the time. Using SunSPOT and some electronic/relays could achieve this.

8. References

[1] Adam Bien, GreenFire. Online: https://greenfire.dev.java.net/.
[2] Solar heating. Online: http://en.wikipedia.org/wiki/Solar_heating.
[3] Paradigma solar heating system. Online: http://www.ulltech.ie/solar-heating.pdf.
[4] GlassFish. Online: https://glassfish.dev.java.net/.
[5] Java Enterprise Edition. Online: http://java.sun.com/javaee/.
[6] Java Standard Edition. Online: http://java.sun.com/javase/.
[7] Project Shoal. Online: https://shoal.dev.java.net/.
[8] Sun SPOT. Online: http://www.sunspotworld.com/.
[9] Java FX. Online: http://java.sun.com/javafx/.
[10] EJB 3. Online: http://java.sun.com/products/ejb/.
[11] FreeTTS. Online: http://freetts.sourceforge.net/docs/index.php.
[12] Sphinx. Online: http://cmusphinx.sourceforge.net/sphinx4/.
[13] Java ME. Online: http://java.sun.com/javame/index.jsp.
[14] p4j5 (patterns for java ee 5). Online: https://p4j5.dev.java.net/.
[15] Franco Zambonelli, Nicholas R. Jennings, Michael Wooldridge, Developing Multiagent Systems: The Gaia Methodology. Online: http://users.ecs.soton.ac.uk/nrj/download-files/tosem.pdf.