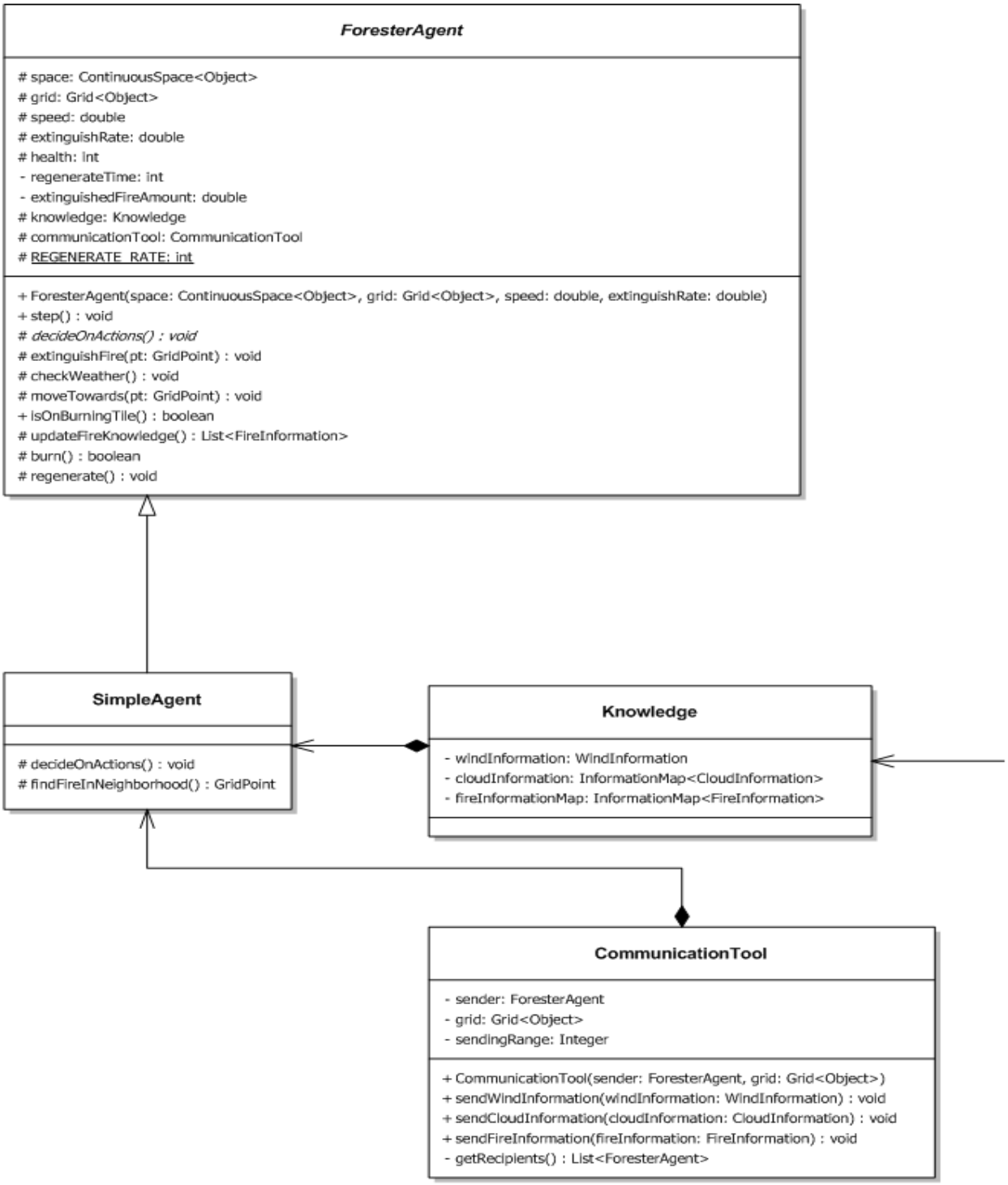
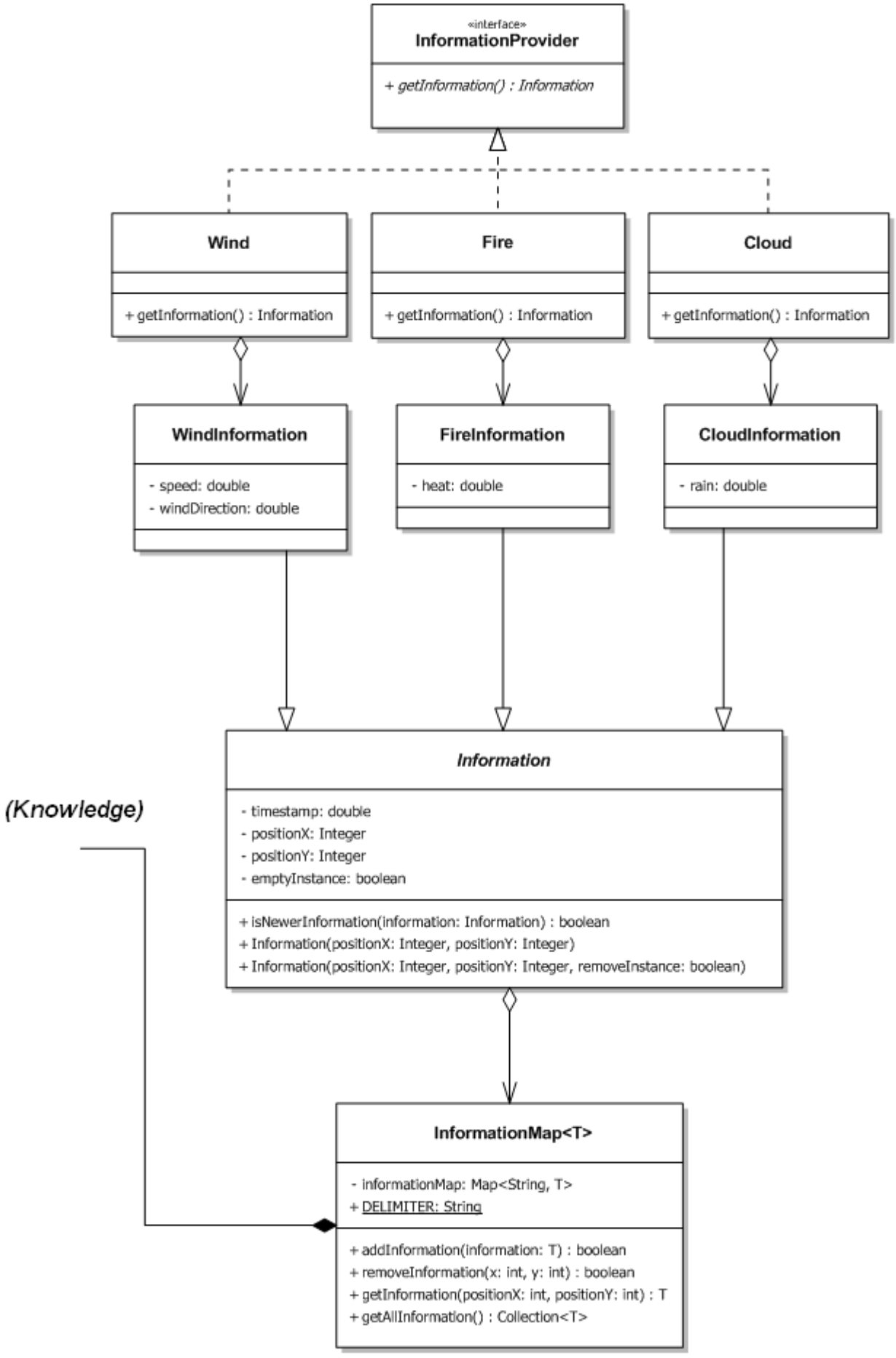
Wildfire Season Report #3
(Simulation Parameters and Result Graphs, Extension to Communication Model)

Multi-Agent Systems Group 6: Daniel Brüggemann, Carsten Orth, Karsten Seelert

Communication and Agent Class Diagram




«interface»
InformationProvider
+ getInformation() : Information
Wind
+ getInformation() : Information
Fire
+ getInformation() : Information
Cloud
+ getInformation() : Information
WindInformation
- speed: double
- windDirection: double
FireInformation
- heat: double
CloudInformation
- rain: double
Information
- timestamp: double
- positionX: Integer
- positionY: Integer
- emptyInstance: boolean
+ isNewerInformation(information: Information) : boolean
+ Information(positionX: Integer, positionY: Integer)
+ Information(positionX: Integer, positionY: Integer, removeInstance: boolean)
(Knowledge)
InformationMap<T>
- informationMap: Map<String, T>
+ DELIMITER: String
+ addInformation(information: T) : boolean
+ removeInformation(x: int, y: int) : boolean
+ getInformation(positionX: int, positionY: int) : T
+ getAllInformation() : Collection<T>

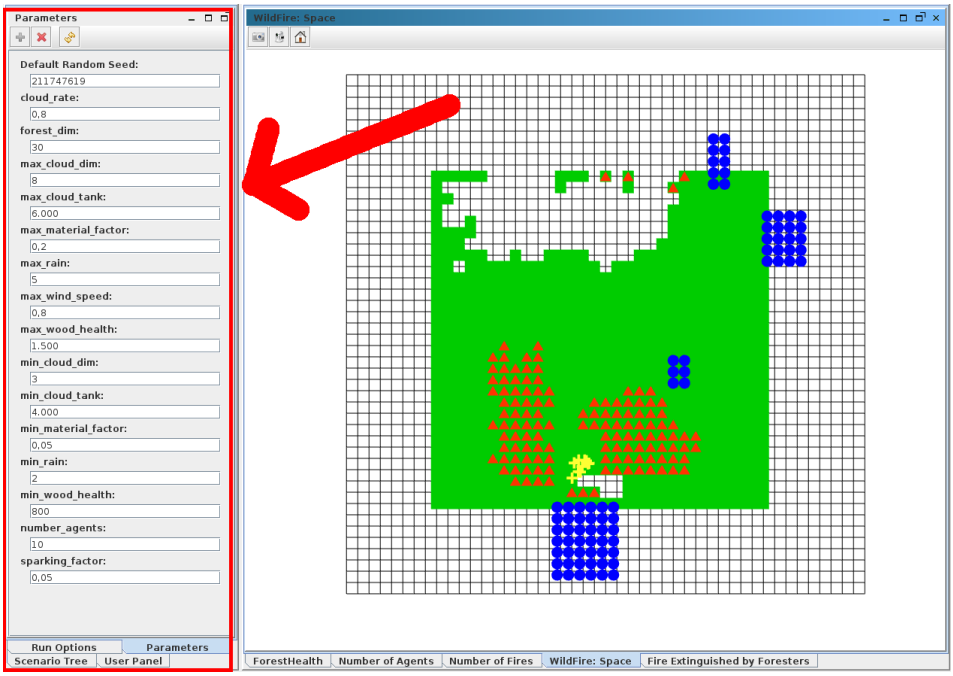
This diagram shows, besides the general agent class structure that was already described in the last report, the classes and interactions related to the communication system. For the classes Wind, Cloud and Fire only the information related method is shown; each of these classes can create a "snapshot" of their relevant data which is stored for certain grid coordinates (can be set null for global information) and a certain time step and represents a piece of information an agent gained at that time step.

Each agent keeps a Knowledge instance which contains several InformationMaps for the different information types (except for the wind which is one global instance). When receiving new information for grid coordinates where such an information type already exists, their timestamps are compared. Older information is overwritten. To also be able to remove information about objects (e.g. a fire which was extinguished), an „empty“ Information object can be send to overwrite old data (emptyInstance = true).

Each agent also keeps an CommunicationTool instance to send information to other agents. Information can be send to all other agents (sendingRange = null) or to agents that are within a specified range around the sender’s coordinates in the grid (which needs less resources). Sending information to one specific recipient, e.g. a group leader, is not implemented yet but planned.

Dynamic Simulation Parameters

In the simulation there are many different parameters which define the environment and the setting of the simulated scenario. In order to keep these parameters variable, we make use of the repast runtime environment. This environment provides a parameter-class, which loads requested parameters from an xml-file. Using this feature we are now able to change some values dynamically in repast's graphical user interface.



The following parameters can be changed right before the scenario initialisation:

forest_dim
This variable defines the squared dimension of our forest.

max_wind_speed
In the simulation there is a global wind, which changes its direction and speed by a gaussian distributed value (lower changes are more likely). In order to limit this wind speed (fire-spreading raises according to wind-speed) you are able to set a maximum speed.

min_cloud_dim and max_cloud_dim
During the simulation rectangled clouds will be generated. The dimension of these clouds is chosen by a random-value between these two values.

min_cloud_tank and max_cloud_tank
Each cloud carries a specific amount of water which decreases over time. If there is no water left, the cloud disappears. A cloud is generated with a random amount of water between these two values.

min_rain and max_rain
In each iteration-step the cloud emits some water to the current cell. Either this water fights the fire within this cell, or the wood gains some wetness. The amount of water, which is spend to the cells changes with a gaussian distributed random value, but it is always inside the given window.

cloud_rate
A CloudFactory generates all clouds with respect to the current wind speed, but the frequency in which clouds will be created also depends on the changeable value "cloud_rate"

max_wood_health and min_wood_health
Each piece of the forest gets some life points. But it is never the case that the whole forest is equal, so each tile gets a random number of life-points within these borders.

max_material_factor and min_material_factor
The material of the wood is defined by a random value between these given values. This factor defines how much water can be stored in this tile, the amount of water it transpires and the behaviour of fire on this wood.

sparking_factor
There is also a `FireFactory`, which randomly puts fire in some cells. A high sparking_factor leads to a higher probability for new fire spots.

number_agents
You can also define the number of agents who have to fight the fire in the simulated forest.

Evaluation Values

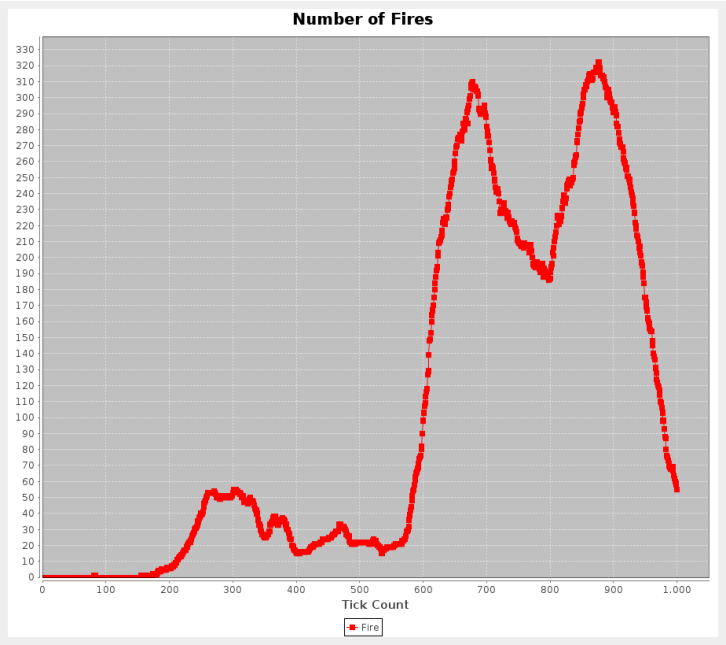
We make use of repast's data-sets and time-series-graphs in order to get some indicating values.
Therefore you have to define the data sources, which will be logged, and the method to aggregate these values.
So far we defined the following items to rate the strategies:

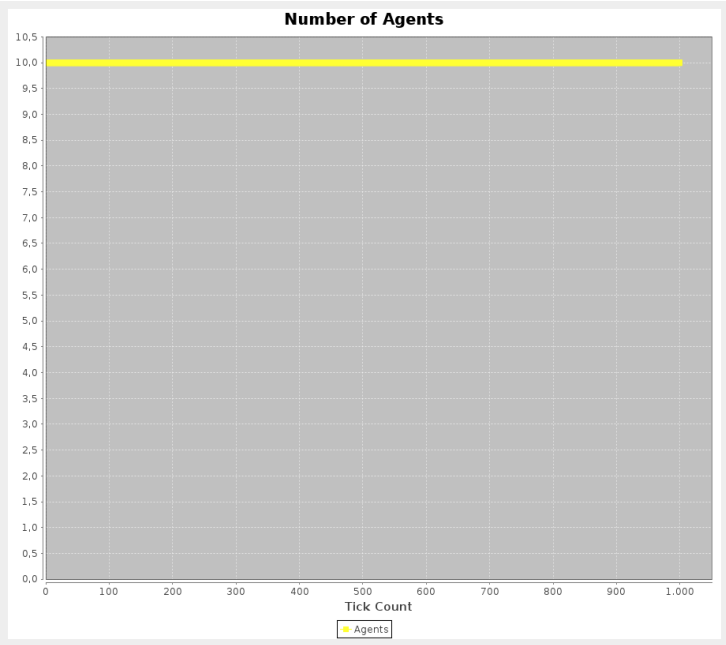
- the number of agents alive
- forest’s health (sum of life-points)

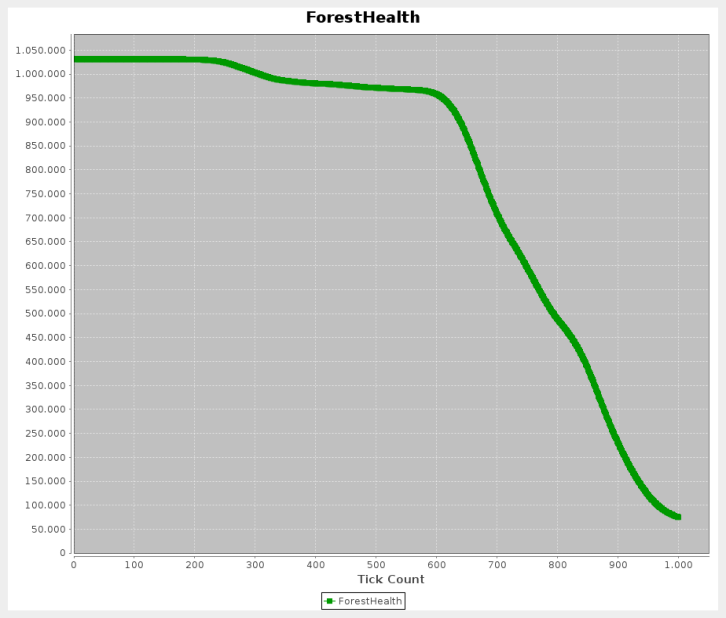
- number of fires extinguished
- (number of fires)

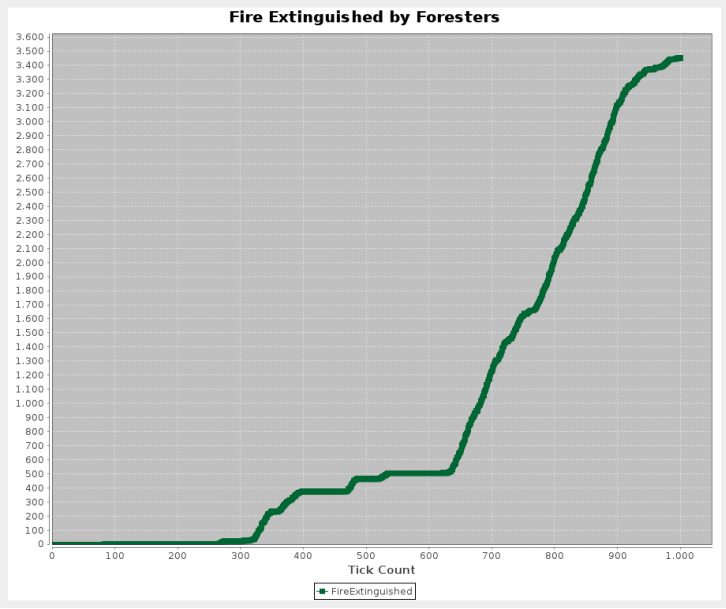
Each agent stores the number of extinguished fires. So if an agent dies, also the number of extinguished fires decreases. But we will change this soon (see *Perspective*).

Repast also provides the opportunity to visualize these data in a graph:









Agent strategies

While still working on a basic level, we now used our communication model to send valuable information between the agents. As of yet, the SimpleAgent's strategy only uses the communication system to send information about seen fire positions to the other

agents. However, this already creates a much more efficient work behavior, because now agents help each other extinguishing fires instead of each one searching on their own. We ran a small test set of 10 simulations until 1000 time steps were reached, once for the agent strategy with and once without sharing information about fire locations. Because the simulation conditions (wind direction, clouds, starting position of fire) influence these values a lot, this shall only serve as an indicator of the value of even simple cooperation patterns. These are the results for the simulation values mentioned above:

	Number of agents alive	Forest health	Amount of fire extinguished
Information sharing	9.4	231000	3529
No information sharing	7	192000	1399

Generally, a lot less fire was extinguished and agents were at a slightly higher risk to die without sharing information about fire locations. In the sharing simulation, agents eventually cluster up at a location and are therefore less likely to die because together, they can extinguish the fire very fast.

Another small addition to the agent strategy was, when the agent is in a burning tile, to flee to burned down wood tiles if possible, because these cannot burn anymore and are therefore save places.

Conclusion and Perspective

The dynamic parameters in the repast UI and graph visualizations allow for more precise tweaking of simulation parameters. It also makes it easier to compare effects of changes in the agent's strategies, which will be a major part of the work in the next week.

The communication system was further developed and can now be incorporated into the strategies. Still missing is an option to send to one specific agent. Also, information about the agent's positions are not yet modelled in the Knowledge and CommunicationTool classes, but are important for example to spread out agents when searching for fires. Because there can be more than one agent on a tile, the usual Mapping system (coordinates -> object) cannot be used. Lastly, requests about a situation (e.g. an agent is trapped and needs help) and answers to these requests also have to be implemented in the information system.

Further TODOs for next week:

- Right now in the graph measuring the sum of extinguished fire over all agents, this sum drops down when an agent dies because his contribution is removed from the context, which is not correct and has to be changed. Every extinguished fire should be considered.
- The "random" movement of an agent when finding no other reason to move to a specific tile is not really random towards all directions, which leads to the agents clustering up mostly in the middle. For better exploration results, this should be corrected.
- The multiple simulation test was done manually; we should look at batch runs to automatize this process if possible.