Zeus Concepts

Document author: John Shepherdson
Document version: 1.0
Document creation date : 18/12/00
Signoff by:
Endorsement list:

Concepts

The following concepts are used to design and construct ZEUS agent applications:

- Agent;
- Goal;
- Task;
- Fact.

Each concept is described in detail below.

Agent

We have identified the generic concepts (as shown in Figure 1) that can be used when specifying Zeus agents:

- Definition - this concept is used to describe the agent's abilities (e.g. can multi-task, can forward plan) and knowledge (which will change with time, as the agent discovers more about its environment).

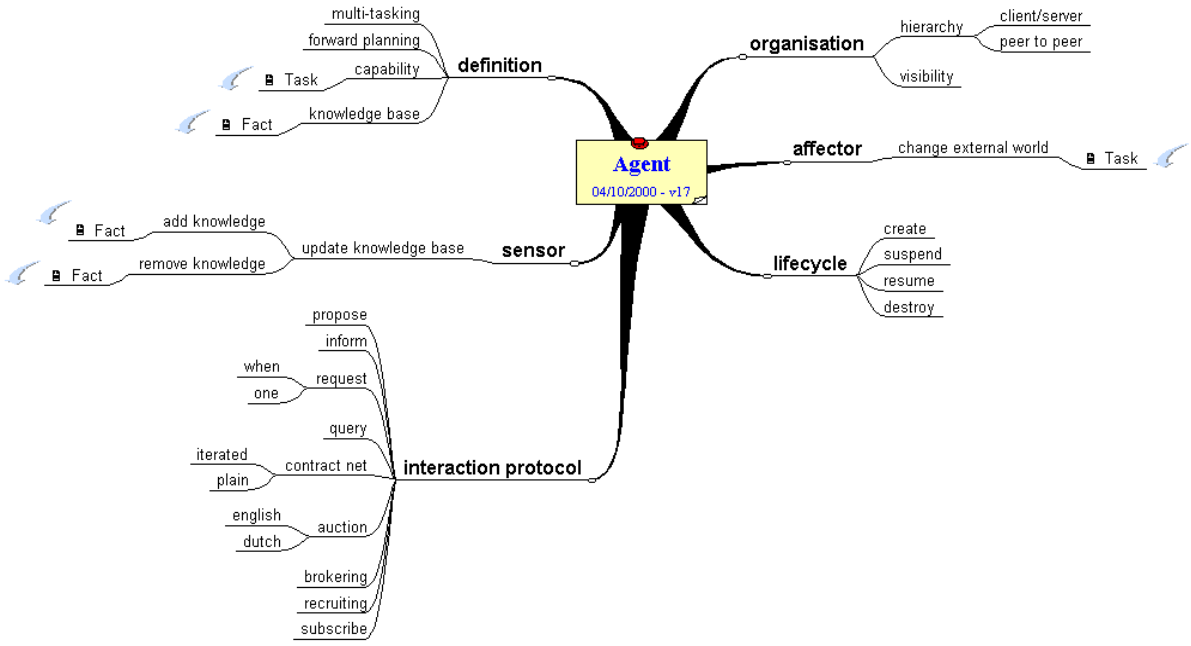

Figure 1: Agent Concept Map

- Organisation - an agent can be ordered by another agent to perform some task, or can collaborate with others to achieve common goals. An agent may be known a priori to other potential collaborators, or may be discovered on an 'as needed' basis via a directory service.
- Sensor - the agent detects changes in its environment via its sensors, and updates its knowledge accordingly.
- Affector (aka Effector) - an agent can change its external environment deliberately, typically as a side-effect of executing some task.
- Lifecycle - defines the valid states of an agent. Initially an agent is created. When alive, an agent can be suspended (to temporarily remove it from the environment), and subsequently resumed. Finally, an agent may be destroyed, in order to remove it permanently.

- Interaction protocol - the agent may possess a variety of interaction protocols, facilitating dialogue with other agents.

Goal

A goal will often be created by an agent with the intention of fulfilling a commitment made to another agent. A goal can be satisfied by performing a particular action. Four important concepts are used to define a Goal, namely constraint, type, motivates and lifecycle (as shown in Figure 2). Constraint defines the earliest start time, latest completion time and latest confirmation time that are acceptable to the entity that set the Goal, as well as the maximum cost and the number of times that the Goal should be invoked. Type defines whether the Goal is one-shot or ongoing. Motivates declares the Task (see Figure 3) that is directly associated with the Goal. A Goal may be created by a postcondition of a Task, as a side effect (*e.g.* of performing a Task), or by some other affector. A Goal may be destroyed as a result of being consumed by a Task (*i.e.* by a Task precondition), by a side effect (*e.g.* of performing a Task), or by some other affector.

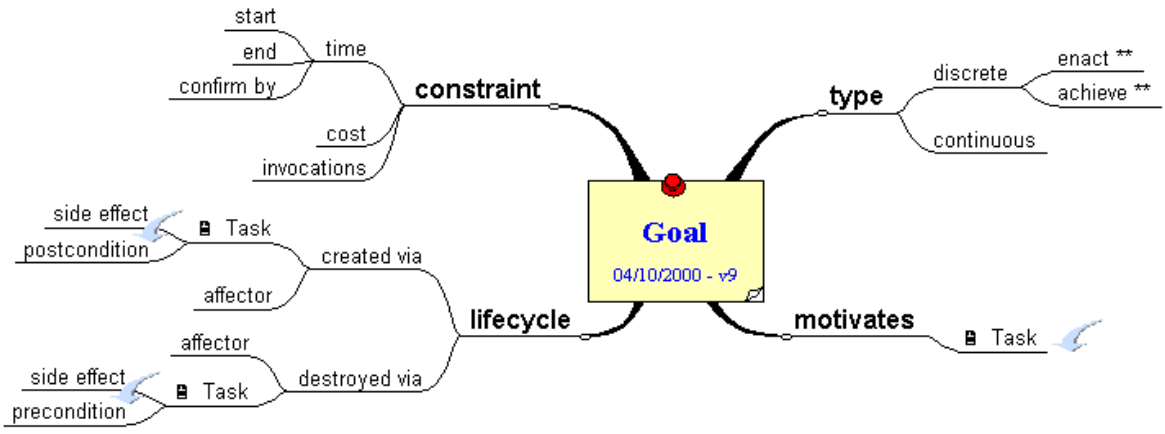

Figure 2: Goal Concept Map

Task

The Task or the Tasks that an agent has to perform to achieve a goal may have some preconditions (as shown in Figure 3). Such preconditions may be based on knowledge in the form of Facts (see Figure 4) retrieved from a database (*e.g.* route planning requires some map information). In some cases it may be necessary to update a knowledge base before starting the Task. Such knowledge can be for instance the actual traffic conditions on a specific road. An update requires the addition or removal (also know as 'consumption') of one or more Facts (*e.g.* a traffic jam now has cleared). Further changes to the

knowledge base may take place as a side effect of executing the code necessary to perform the Task. A Task may also produce Facts upon completion, thus updating the agent's knowledge yet again.

Constraints are special requirements which build a link between a Task's pre- and post-conditions. For example, the number of items (such as work requests) present after Task execution must equal the number available before Task execution.

In some environments it may be necessary to decompose the Task into autonomous sub-Tasks. In all other cases a Task is atomic. Every Task has attributes (such as cost and duration) that provide an agent with information needed for planning and decision-making purposes. Each Task is assigned to one agent - as task owner, that agent has responsibilities such as starting, controlling and cancelling the Task.

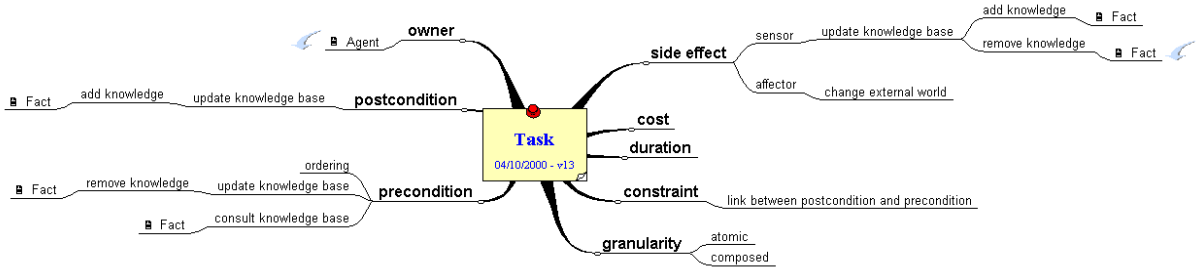

Figure 3: Task Concept Map

Fact

A Fact represents something that an agent believes to be true, either about itself or its external environment (*e.g.* an agent may believe that it has the name "ZeusAgent004" or that there is heavy traffic between junctions 24 and 26 of the M1.) A Fact has a number of characteristics: lifecycle, visibility, constraint, owner, attribute and type (as shown in Figure 4). A Fact may be created upon completion of a Task, as a side effect (*e.g.* of performing a Task), or by some other affector. A Fact may be destroyed as a result of being consumed by a Task, by a side effect (*e.g.* of performing a Task), or by some other affector. The visibility of a Fact determines its accessibility. Constraints can be applied to its attributes (*e.g.* the value of attribute A is equal to three time the value of attribute B). Owner identifies the Agent that the Fact belongs to. A Fact has two types of attributes, those that are always present (*i.e.* standard attributes), and those that are domain dependent (*i.e.* custom attributes). The value of a Fact may be either deterministic (*i.e.* fixed) or non-deterministic (*i.e.* variable). The type of a Fact determines whether it is consumed by a precondition of a Task, or merely referred to.

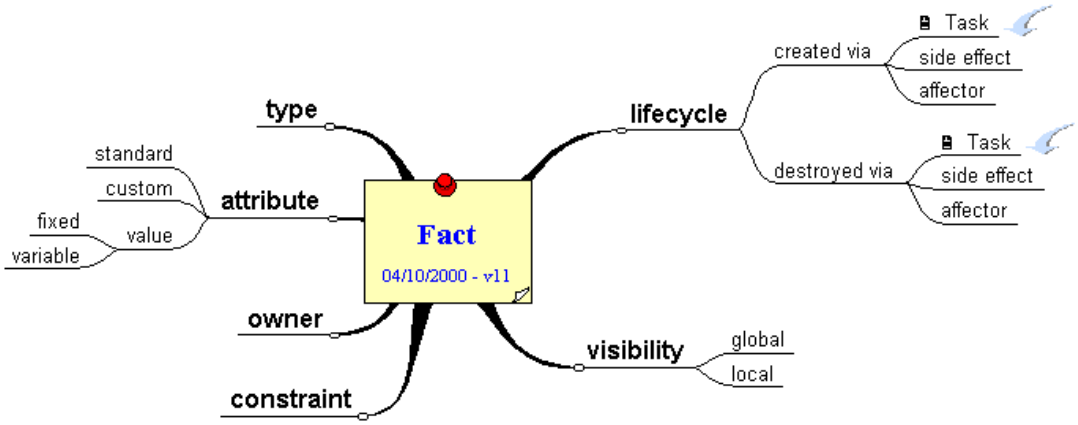

Figure 4: Fact Concept Map