Towards the Necessity of Constructivism as an Epistemology of Agents in the Social Sciences

J. Woch
Institute for Computerlinguistics
University of Koblenz
Rheinau 1
D-56075 Koblenz
email: woch@uni-koblenz.de

Abstract

It turns out that models of social agents are often based on wrong assumptions in the fields of cognition and epistemology so far. This regularly leads to inadequate models of social systems and its behaviors, especially those of communicative nature (planning, negotiating, inspiring confidence etc.). The constructivism seems to provide an appropriate principle for the modelling of inner and outer worlds, because of its refusal of the existence of mutually knowledgeable truth. Therefore, agents based on the principles of constructivism are much better situated for the modelling of social systems.

After a discussion of the formal background of constructivism, an architecture of such agents is shown which allows to model constructivistic behavior. Practical issues of the architecture are also considered, as well as work, which is left to the modeller. After this, a short example is given to show the advantages and drawbacks of this approach.

1 Do we need constructivistic agents?

Under the aspect of the complexity of interaction with their environment, agents may be distinguished according to the definition of [Sundermeyer, 1993] in the following manner:

Reactive agents are able to react to environmental stimuli by a response dictated by a more or less complex rulebase. We can consider them as deterministic finite automata. They are able to operate in a deterministic environment, but stimuli not considered in the rulebase are effectless, or worse, not even perceptible by those agents. They are at the mercy of the elements. Therefore, reactive agents need no knowledge of their environment.

Intentional (or BDI) agents are able to have goals and plans. They may set priorities between these goals in dependence on their motivations. They are not independent of the environment and its stimuli, they may however make their choices between the alternatives given by different goals, and – more important – they are able to take action without the need of being stimulated by the environment. Therefore, intentional agents require a representation of the environment, but not necessarily the representation of the their own goals or, more generally, a representation of Self.

Social agents take special care about their co-agents in the environment, therefore they are considered as derivations from intentional agents. They try to create representations of the co-agents' goals and, being aware of their own communicative capabilities, may try to instrumentalize them in order to reach their own goals. Therefore, social agents necessarily require a representation of Self. This is somewhat recursive, because a representation of Self can contain a representation of Self and so on.

The way an agent perceives the environment or co-agents in BDI-architectures is crucial to the quality and complexity of the knowledge, and vice versa: the knowledge about its environment is crucial to the way an agent perceives. Most commonly the observations or results of ongoing perceptions are treated as secure and reliable. Worse, it is assumed that there is no distortion, no interpretation of that information. In order to model confidence or uncertainty modellers then implement reasoning mechanisms, which judge the newly arriving information in the light of the already known. They deal with the reliability of communication by distinguishing between obviously true (or false) and possibly true (or false) information according to the agents' confidence in the sources. While this is sometimes necessary for the modelling of trust in social communities, it is unnatural and does not meet the point: It assumes, that there *is simple discernible and communicable truth.* This assumption is correct for scenarios in which internal states aren't part of the communication process. Unfortunately, even if the perceptional process itself is secure and reliable, it is wrong in social communication scenarios. Here it is very important, how the interpretation process is influenced and in which way the representation is built. So, the modeller should consider to eliminate this kind of truth in social models and constructivism is a promising approach:

What does an observer perceive, when he observes? Often, different observations of the same entity or event lead to different realizations. This results from the fact, that different observers have different experiences and therefore a realization is understandable in conjunction with the observer only. By assuming a common physical world it is still usual to find uncommon views of the same world. This is a consequence of the postulate 1.1

Postulate 1.1 *Living systems are structurally determined, i.e. their state changes are completely dependent on their internal structure [Maturana and Varela, 1990].*

Therefore, a stimulus may trigger a state change, but never determines it. This happens through translation of the stimulus into an internal representation – on the contrary to DFA's, where no translations take place. This translation decouples the system from its environment with the following effects: First, it is ensured that a system is able to recognize the bare existence of a stimulus. Second, the system's state itself may stimulate the system. So it is possible to trigger a cascade of state changes.

A first (and most important) conclusion is: There's no way to distinguish between truth and deception. Deception even takes place if the observer is aware of the deception, in other words when he 'knows the truth' (for the unbelieving: Please cross two fingers of your hand and fumble for your nose. Don't you have two?). This immediately leads to:

Postulate 1.2 *The living systems' representation of the environment is inherently uncertain.*

Structural determination leads to a second aspect: If a state change of a system should be triggered, but not be determined by the environment, the state function f (not to confuse with translation!) has to be defined $f : s \to s$, where s is the state space of the system. This means, f discloses the environmental state space from its 'considerations'. Therefore, f is restricted to its state space by definition, and a system behaving in the described manner is called operationally closed. This closeness is the "principle, which makes cognition a constructive process" [Hejl, 1991]. We should state here that operationally closed systems always construct representations of their environment self-referentially, i.e. based on their experiences. They are viewing the world through their glasses of former knowledge.

Agents who do not behave in the described fashion, i.e. have no 'cognition problem' clearly are not able to (learn to) deal with it. Models assisting in the investigation of the genesis of social systems, language, or in dealing with contingency (a consequence of postulate 1.2, s. [Luhmann, 1994]) can definitely not be done appropriately using non-constructivistic or 'naive-epistemological' agents. Therefore, we'll provide an architecture which inherently allows for constructivistic behavior of agents, without restricting too much conceptual decisions in other fields of agent modelling. However, the job of developing an appropriate meaning generating operation is left to the modeller.

2 The architecture

Fig. 1 shows a schematic representation of a 'constructivistic architecture' (GENESIS). Incoming (as well as outgoing) messages are processed 'physically' by Filters, whose operations are parametrized by Properties and Abilities. Then they will be delegated to Behaviors, organized as tasks in a Scheduler. Behaviors process incoming messages 'mentally' by operating on a mental state only, thereby generating and processing a symbolic representation. Additionally, they take action for the agent by generating messages and sending them off.

Now we will give a short overview of its elements (for details see [Woch, 1997]) and then focus on the aspects relevant to our discussion.

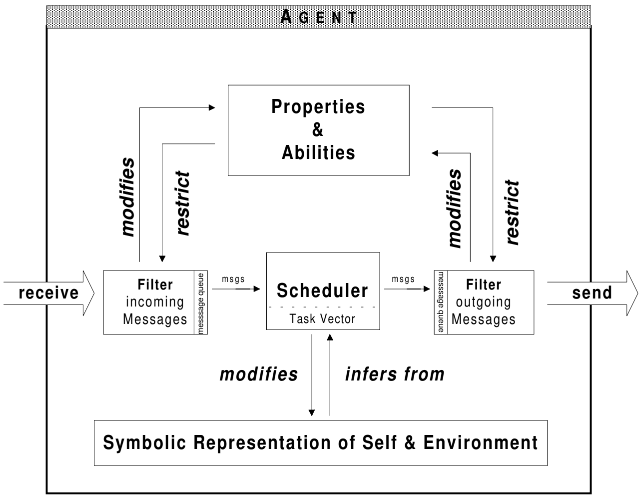

Figure 1: Overview of the architecture

The aim of GENESIS is twofold: First, it ensures the constructivistic approach. For that, the overall principle of GENESIS is based on two columns: The 'physical' aspect of agentship as usual is modelled by a) the properties and abilities and b) the filters of incoming and outgoing messages. The 'mental' aspect of agentship is modelled by the representation of self

& environment (SR).

However, the 'physical' and 'mental' state spaces are strictly separated from each other: The filters only operate on 'physical' structures, the tasks only on 'mental' structures.

The second aim is to allow widely conceptual decisions in the fields of believing, planning, decision making, coordinating and communicating – as long as it doesn't injure the first aim. For that, the object model of GENESIS consists of compulsory and facultative elements, called parts and components. Roughly speaking, the parts establish an agent's structure, whereas components may be viewed as enrichments of an agent's state space and room for manoeuvre.

2.1 Parts ...

An agent consists of the parts shown in fig. 1. Each part, except for the filters, provides a container for its specific components. An interface provides read- and write-access, restricted to the corresponding component-classes. The storage for the components is implemented as a hashtable, using the name of the object to be stored as key. Hence, two objects with the same name cannot be stored in the same container.

Symbolic representation (SR)
The SR contains everything an agent considers worth enough to memorise. It is implemented as an embedded "prolog"-object, which has several advantages:

- Modellers are able to apply their experiences in the fields of logic and AI programming.
- There already exists logic-based solutions for common AI-problems, which easily can be adapted instead of re-inventing the wheel.

Beside others, the mental representation of goals and plans is a common usage for the SR. However, it is the matter of modellers which and how goals, plans or beliefs should be stored, how and when an agent can reason about them and what to do with the results (for example, one may use the specification of beliefs described in [Benerecetti *et al.*, 1997]).

Surely, agents may believe, that they've these or those goals or plans (or beliefs at all – this may recurse into infinity) and even may act respectively.

This assumption is correct for scenarios in which internal states aren't part of the

Scheduler
The scheduler is a container for `Tasks`. In GENESIS `Tasks` are threads, hence they've the possibility to run concurrently. There is no scheduling algorithm given by default: Inhibition, promotion, starting and ending of tasks is a matter of the tasks themselves. So the modeller may choose cooperative or uncooperative, concurrent or sequential task scheduling. He could even create a `Task`, whose job is the scheduling of all the other tasks, thus creating a meta-`Task`, or he could have the tasks do the handshaking themselves.

Filters
Filters in some way are the sensors and effectors of agents, however, not generally: The 'sensors' take massive effect on the properties and abilities of its agent, so these names would be inappropriate. Filters are responsible for the evaluation of the active and passive definitions of abilities carried by messages (see below).

The receiver is not able to ward off a message; its constraints are always checked, possibly leading to the execution of the ability definitions. Therefore, filters can be viewed as the 'physical case' of an agent. Additionally, incoming messages are always put into a queue, in which they wait until noticed by any task (at least, if there is a task that is interested in incoming messages). This queue is necessary for a facultative cognitive reaction to the messages – otherwise an agent wouldn't be able to be aware of them.

2.2 ... and components

Every agent is an instance of the same class, i.e. different agent types differ only in the setting of its components. The advantage is obvious: New agent species may be created during simulation.

The components of this architecture are exchangeable, addable and removable during lifetime, i.e. agents are able to perform modifications to the dimensions of their physical state space, to gain (or loose) abilities, behaviors and even properties.

Components are named, which are, being symbolic pointers, ambiguous: two different objects may have the same name. This is intentional, because the exchange of 'hard links' would violate the principle of self-reference. However, by this way we've introduced misunderstandings.

Properties
Properties describe the 'physical' state space of agents. They quantify the abilities and are modified by them.

Often the change of the value of properties should cause the change of other properties. To enable this sort of 'chemical reaction chain' within a given agent, properties may change the values of other properties without stepping through the filters.

Properties may be built of arbitrary complexity.

Abilities
The definition of abilities describes which action an agent can perform, and what happens, if he does so. An ability is fully defined by

$$\text{¡}alias\text{¿:¡}spec\text{¿}\left[\frac{condition}{action}\right]_A \left[\frac{condition}{action}\right]_P sync$$

where ¡*alias*¿ is its name, ¡*spec*¿ an arbitrary parameter, $\left[\frac{condition}{action}\right]_A$ the conditions and consequences for the agent, if he is the actor (active part), and $\left[\frac{condition}{action}\right]_P$ the conditions and consequences for the agent, if he is the target (passive part).

For an agent it is definitively impossible to predict what happens to its target, because this depends on the target's own definition. Therefore, after activating an ability, an agent ought to examine the results.

Activation of abilities always is performed by the filters, who extract the `Ability`-names from incoming and outgoing messages, which in turn are generated by Tasks.

Tasks

Tasks implement behavior. Generally, they are the *omnia movens* by reasoning and taking action. Some examples should illustrate this:

- A task **harvesting** could be responsible for harvesting food
- a task **coordinating** could be responsible for the coordination of collective goals
- a task **planning** could be responsible for the creation of plans to achieve individual goals

For that, tasks make agents 'active'. However, agents without any tasks are, given appropriate properties, at least reactive.

As discussed above, tasks are computed concurrently. According to the operational closeness of agents, tasks are only allowed to access mental structures, i.e. the `SR` and the `MessageQueue` (which also can be seen as a kind of memory). The latter is necessary to be able to cognitively react to events. The scheduling of tasks is described in section 2.1.

The environment

The environment of a model in GENESIS has four purposes:

First, it provides a common surrounding for all agents.
Therefore, it has properties which describe the 'physical existence' of the world, e.g. the topology.

Second, it provides 'laws of nature', i.e. a specific, deterministic and reliable behavior.
Therefore, it has its own tasks (`WorldTasks`), just as agents have.

Third, it provides communication capabilities for the agents embedded.

Fourth, it allows for multi-layer-modelling techniques, i.e. the hierarchical cascading of worlds.
Therefore, it can be owned by agents as a property, i.e. it itself is a property.

Messages

Actions taken are transported via messages. A message m transports information I from sender S to receiver R:

$S \xrightarrow{I} R$, where S and R are members of the same world.

The transportation is not peer-to-peer, but directed through the world: $S \xrightarrow{S \xrightarrow{I} R} World \xrightarrow{S \xrightarrow{I} R} R$

Therefore, a message could fail, because R doesn't exist, or R isn't the agent intended by S.
I always transports `Abilities`.

3 Modelling with GENESIS

Generally, all the agents in a given world are instances of the class `Agent` – they only differ in the settings of their components. Therefore, the class hierarchy in GENESIS is definitely not equivalent to the world's intended ontology: A given world may contain dogs and cats, but they are all represented by instances of the same class. So you do not need to create a new class for a new species. Instead, get a new instance of `Agent` and endow a dog-in-spe with dog-like features and a cat-in-spe with cat-like features. With respect to the fact that each component is potentially addable or subtractable, an agent is not necessarily tied up to its current species and may more or less subtle 'mutate'.

This is a consequence of the component-based architecture relevant to the model's behavior. Another consequence relevant to the modelling process is that the modeller creates new agent species by assembling the needed components (provided by a component library) to a 'skeleton', initially instantiated from `Agent`. He then could store such schematic species into a schema library for later use – or duplicate this or other schemata (provided by a schema library) as often as necessary for the intended model.

The modelling process with GENESIS consists of three layers. These layers should be understood as guides through different engineering processes by breaking down the whole modelling task in a conceptually more understandable and tool-assistable manner.

- At the **implementation layer** modellers create new component classes by deriving them from the abstract component classes.
- At the **schema layer** modellers create new species by using existing components.
- On the **model layer** modellers create new worlds by using existing species.

Finally, after modelling, there's a fourth layer, the **simulation layer** in which modellers set initial parameters and run the simulation.

Of course, the implementation layer covers most of the work, especially the implementation of tasks. This components describe the behavior of agents and are very often model specific, which seldom can be recycled without modifications. Despite the necessity of a general understanding in programming, the modeller is likely to experience the limits of his 'power' over the agents: An agent or one of its behaviors is not able to 'read' or 'set' a value of a property, just as he wishes. Each change of the physical state, even an observation, whether of itself or of others, necessitates an appropriate action – and if the agent is not able to do so, the modeller is not able either.
Another point related to tasks should be kept in mind. Since tasks potentially operate concurrently, the modeller is likely to encounter common resource conflicts, not only between agents, but within a single agent too.

Things are going to be easier at the schema layer. At this stage the assembling of components to a single agent schema takes place. Provided with a rich component library, modellers can focus on the overall behavior of agents in an individualistic point of view. This step can be supported by sophisticated tools, as outlined in section 4. The results may be stored in a schema library for later (re)use.

The next abstraction leads to the model layer, at which modellers assemble instances of schemata, set their individual values of attributes and describe the environment, in which the agents will be embedded. At the model layer modellers focus on the overall behavior of their model from a collective phenomena or social systems's point of view. This step, too, can be supported by sophisticated and intuitive tools – a very important advancement for the educational use of GENESIS.

At the simulation layer modellers modify simulation parameters of special interest (birth rate, metabolism, confidence, whatsoever) and run the simulation. Possibly visualizing the results may follow the simulation layer.

As in software-engineering processes stepping through the four layers usually is a cyclic process.

3.1 An example

The basic goal of this example is to examine the influences of collective misbelief on the population growth. It is not always the case that collective misbelief must be harmful to a society of agents. "To take a human example, consider a group which (incorrectly) believes that a certain stream is holy so that water may not be taken from it. This may be to their detriment. But it may also be to their benefit if the stream is dangerously contaminated in some non-obvious way." ([Doran, 1998]).

To experiment with this hypothesis, consider the model discussed in [Doran, 1998]. Generally, a collection of agents is harvesting resources, which are

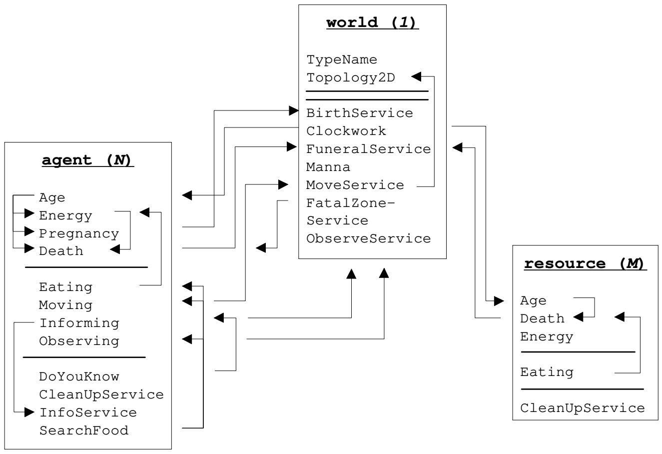

Figure 2: Components for 'Simulating collective Misbelief'

spreaded throughout a plane. Somewhere a fatal zone lurks for entering agents, killing them immediately. The agents are aware of the current co-agents and, additionally, they may believe in the existence of a small number of agents that do not exist in the environment (pseudo-agents). Sometimes they tend to harmonize their beliefs. Offsprings inherit the beliefs of their parents, too.

Agents do not try to harvest resources, when some other agent is better placed to harvest them.

These assumptions probably lead to a particular (mis)belief, which is spreaded throughout the population. Presumption: Agents with a misbelief which prevents it to enter the fatal zone have a better survival probability and therefore have a better probability to reproduce and to share their belief. The population grows.

Where GENESIS steps in
Implementing the above model with GENESIS leads to the structure shown in fig. 2. There are three schemata `Agent`, `Resource` and `World` with the cardinality N, M and 1. The top section of a schema describes `Properties`, the middle section `Abilities` and the bottom section `Tasks`. Arrows outside of a schemabox describe communication via messages, inside of a box direct manipulation. For example, the `Resource`'s property `Age` directly manipulates `Death` which in turn is checked by `World`'s `FuneralService`.[1]

This example does not illustrate all of the architecture's aspects (e.g. multi-level modelling, sophisticated communications), though, it demonstrates the importance of appropriate meaning generation by (deliberately) specifying an erroneous mental operation, which mistakenly adds non-existent agents to the men-

[1] A JAVA-implementation can be examined at `http://www.uni-koblenz.de/~woch/projects/genesis/`

tal representation. This is a kind of world construction whose variation from the "reality" is never (and in this case: should never) discoverable by agents.

4 Conclusions and future work

It has been shown, with respect to epistemicological issues, that architectures which provide a representation of Self without selfreferential meaning-generating operations, make unnatural assumptions, especially if designed for simulation in the social sciences (e.g., using Searle's speech act theory as a basis for communications as in [Dignum and van Linder, 1997], where authority relations between agents specify the authority needed for a speech act to take effect). However, it should be stated that these concepts have their own rights under some circumstances – whether they are given or not is heavily dependent on the aims of the model.

On the other side, the consequences of the constructivistic approach, i.e. the need to validate their observations and even their actions of agents and the need of acquiring each single piece of knowledge about their environment and even about themselves by observations could lead to situations, in which the effort for modelling the environment could prevent agents from doing their jobs.

But the backgrounds of these 'drawbacks' are the hidden strength of the system: Only now it is possible to model processes of understanding on an individual or collective level – a basic stipulation for the investigation of social processes.

Some of the advantages of component-based modelling should be clear: The components very often may be reused without modifications in further models, and they are pluggable at runtime, i.e. agents are able to get new abilities or other components (there must exist such components, however), or loose it after a while. Finally, because of their pluggability, components are candidates for graphical modelling environments.

Currently, we're going to reimplement the component-concept using JAVA-BEANS (s. [Hamilton, 1997]). Doing so, we gain an enhanced interoperability as well as the possibility to use existing Bean-editors (e.g. the `BeanBox` from SUN) for a graphical modelling process at the schema- and model-layer, possibly like shown in fig. 2 – a much more intuitive approach than fumbling with JAVA-code. Finally, using BEANS enables the distribution of agents over a network, thus allowing semi-mobile agents.

References

[Benerecetti *et al.*, 1997] Massimo Benerecetti, Alessandro Cimatti, Enrico Giunchiglia, Fausto Giunchiglia, and Luciano Serafini. Formal Specification of Beliefs in Multi-Agent-Systems. In Müller et al. [1997].

[Dignum and van Linder, 1997] Frank Dignum and Bernd van Linder. Modelling social agents: Communication as action. In Müller et al. [1997].

[Doran, 1998] Jim Doran. Simulating collective misbelief. *Journal of Artificial Societies and Social Simulation*, 1(1), 1998. http://www.soc.surrey.ac.uk/JASSS/1/1/3.html.

[Hamilton, 1997] Graham Hamilton. JavaBeans API Specification. Technical report, SUN MicroSystems, 1997. Located at http://java.sun.com/beans.

[Hejl, 1991] Peter M. Hejl. *Konstruktion der sozialen Konstruktion: Grundlinien einer konstruktivistischen Sozialtheorie*, pages 303–339. In Schmidt [1991], 4. edition, 1991. (First published 1987).

[Luhmann, 1994] Niklas Luhmann. *Soziale Systeme: Grundriß einer allgemeinen Theorie.* Suhrkamp Taschenbuch Verlag, Frankfurt am Main, 5. edition, 1994. (First published 1984, Suhrkamp).

[Maturana and Varela, 1990] Humberto Maturana and Francisco Varela. *Der Baum der Erkenntnis.* Goldmann Verlag, München, 1990. (First published 1984, *El arbol del concocimiento*).

[Müller *et al.*, 1997] Jörg P. Müller, Michael J. Wooldridge, and Nicholas R. Jennings, editors. *Intelligent Agents III. Agent Theories, Architectures and Languages*, number 1193 in Lecture Notes in Artificial Intelligence, Berlin, Heidelberg, New York, London, Paris, Tokyo, Hong Kong, 1997. ECAI'96 Workshop (ATAL), Springer-Verlag.

[Müller, 1993] Jürgen Müller, editor. *Verteilte künstliche Intelligenz. Methoden und Anwendungen.* BI Wissenschafts-Verlag, Mannheim, Wien, Zürich, 1993.

[Schmidt, 1991] Siegfried J. Schmidt, editor. *Der Diskurs des Radikalen Konstruktivismus.* Suhrkamp Taschenbuch Verlag, Frankfurt am Main, 4. edition, 1991. (First published 1987).

[Sundermeyer, 1993] K. Sundermeyer. *Modellierung von Agentensystemen*, chapter 2.1 - 2.6. In Müller [1993], 1993.

[Woch, 1997] Jens Woch. A component-based and constructivistic Agent Architecture for the modelling of MAS in the Social Sciences. Fachberichte Informatik 31–97, Universität Koblenz-Landau, Institut für Computerlinguistik, Rheinau 1, D-56075 Koblenz, 1997.