

# Agents with non-anthropomorphic lifecycles

Ladislau Bölöni, Paul DeJung and Damla Turgut

Department of Electrical and Computer Engineering

University of Central Florida

Orlando, Florida, 32816

Email: lboloni@cpe.ucf.edu, dejongp@saic.com, turgut@cpe.ucf.edu

## Abstract

Most software agent lifecycles are designed in such a way that they mimic the lifecycles of humans (anthropomorphism). This position paper presents a set of operations for software agents which have no immediate biological analogies, such as splitting and merging agents. We argue through a series of scenarios the potential benefits of this approach. We present a method to implement these operations in the Bond agent framework.

## Introduction

Many of the operations of software agents reflect an anthropomorphic view of agents. Operations such as startup and termination correspond to the human cycles of birth and death, learning is usually understood similarly as in the human learning process, the message based communications are explicitly inspired by Searle's speech acts theory of human communication.

On the other hand, we can implement operations such as cloning and splitting an agent, merging two agents or mutating them during runtime. These operations do not have immediate correspondence in the lifecycles of humans, and therefore were less explored. For instance, a survey of literature will find only about twenty articles dealing with cloning agents, as opposed to hundreds concerning agent mobility - despite the fact that agent mobility is normally implemented through cloning.

We will call anthropomorphic the operations of agents (or software in general) which were inspired from the behavior of biological (typically human) agents. Let us review these operations:

- **Startup.** The agent is created and a new identity is assigned to it.
- **Termination.** The agent is terminated. It will not perform any future actions, and any further attempt to communicate with the agent is unsuccessful. The agent framework usually tries to assure that the unique identifier of the agent is not reused.
- **Hibernation.** The agents' operation is temporarily suspended. Messages sent to the agent may or may not be

queued for later reading. At the end of hibernation the agent resumes with the same identity.

- **Message based communications.** Most agent communication languages such as FIPA ACL (Foundation for Intelligent Physical Agents 1997) or KQML (Finin *et al.* 1994) have their roots in the theory of speech acts.
- **Mobility.** It is designed to emulate the mobility of humans in the physical world. The agent continues with the same identity in a different host.

We will call non-anthropomorphic operations which are not direct mappings of biological events. Some examples of these operations are as follows:

- **Cloning / Splitting / Fission.** An existing, running agent is duplicated in such a way that the resulting agents are either largely identical to the original agent (cloning) (Shehory *et al.* 1998) or they are different, but their union contains all the parts of the original agent (splitting or fission). At least one of the new agents needs to acquire a new identity.
- **Merging.** Two existing agents are merged into a single new agent. There are various choices regarding the identity of the resulting agent.
- **Half-life.** An agent maintains its identity and partial communication capabilities after termination.
- **Mutation.** The agent is radically changed during its execution time, while still maintaining its nominal identity.
- **Transplant.** The identity of an agent is transplanted into the physical or virtual incarnation of another agent.

In the remainder of this paper, we will investigate the relevance and utility of non-anthropomorphic operations and propose an implementation approach.

## Relevance of non-anthropomorphic operations

Many technically possible operations can be dismissed as simple implementation artifices, without real world significance. We argue however, that the non-anthropomorphic lifecycle (NALC) operations, as presented above, have relevance in the theory and practice of agents.

Some NALCs are more natural to the virtual world in which the agents live than the corresponding anthropomorphic operations. While mobility in the physical space is

a natural property of solid objects, mobility in the virtual world is a complex operation, governed by different rules and failure models.

Some of the NALC's do have real world counterparts, although not in the domain of naive biology. Splitting and merging are very natural operations in the lifecycle of organizations. Companies split and merge with complex identity transfer operations. A military platoon can divide into independent subunits for a mission and reunite later.

In short, what can we expect from agents with non-anthropomorphic lifecycles? First, different interaction patterns, both among agents and with human users. Users will need to get accustomed to the notions of agents splitting, merging and mutating. It is difficult to assess how steep this learning curve will be. Interaction patterns very different from anything found in the physical reality, such as peer to peer file sharing, became widely adopted, while attempts to accurately model the physical reality in the virtual world, such as the user interfaces based on the "living-room" metaphor, were met with indifference.

We can also expect new kinds of emergent behavior of the agent societies using non-anthropomorphic lifecycles. For example, algorithms inspired on the behavior of social insects such as ants were proposed for problems such as path optimization, routing etc. These models however, exploit only the mobility operation and ant-agents with a single pre-programmed behavior. Adding additional lifecycle operations to the genetic pool" we can obtain a new, richer set of emergent properties.

Finally, we need to clarify the relationship of the operations we propose with genetic algorithms and evolutionary programming. The terms of cloning (reproduction), mutation and one particular type of merging (crossover) are used for long time and considerable success in these fields. These operations, however, are considered in the context of reproduction as genotype operations, not as operations in the lifecycle of a single agent, as proposed in this paper.

## Real world interpretations

Many NALC scenarios can be given consistent real-world interpretations. These interpretations can be immediate applications of agents with NALC operations. In this section we will give two real world examples which lead naturally to an agent model using NALCs.

There are however other situations where the real-world analogy is misleading. To illustrate this, we will also give an anti-example.

**Example 1: Corporate split and merge.** Splitting and merging is a frequent event in the life of corporations. In case of a corporate merger, the identity of the previous corporations need to be carefully managed. In some occasions, both corporate identities need to be maintained. In other situations, one of the identities will be completely absorbed in the other and in yet other cases, a new merged identity is created.

Corporate entities, despite their complex internal structure, can be modelled by a single agent - in fact, they are treated legally as such. The merging of corporate enti-

ties creates similar problems with the merging of agents. The "knowledgebases", goals and plans of the two entities need to be merged and potential conflicts resolved. The resources, commitments and responsibilities of the merged entities must be overtaken by the new one.

Similarly good analogies can be found for the splitting of corporations. Some of the assets of the corporation are partitioned to the new entities, while others, such as "know how" might be replicated.

If agents are used in simulations representing corporate entities, the parallels are immediate. But even beyond the field of simulation, agents have many similarities with corporate entities. Both are complex structures, with a set of goals, knowledge and capabilities. They are owners of resources, have commitments and responsibilities and are subject of the laws of their environments. The corporate laws governing the merging and splitting can serve as an inspiration for the laws of agent societies.

**Example 2: Military units.** Military units are frequently split into subunits in order to accomplish their mission. Scout units are split from the main unit and sent to investigate the terrain. Subunits remain behind to secure the supply lines. Units are split to escort prisoners, build fortifications. Certain subunits might receive completely new orders, in which their goals, organizational structure and capabilities change radically - the equivalent of agent reconfiguration/mutation. The mechanisms for these operations have similarities with the ones for corporate entities, but they also have specific differences. Military subunits can be lost as a result of enemy action, but their goals are maintained and transferred to other units. Splitting, merging and reconfiguration of military units happen on a much faster timescale than the corporate reorganizations. An additional characteristic of the operations of the military units is the *chain of command*, which is maintained throughout these interactions.

Just as with corporate entities, simulations of military units can benefit from the operations of reconfiguration, splitting and merging. Furthermore, the clear chain of command of military units is a better match for the situations where agents respond to the goals set forth by a single user.

**Antiexample: Exploring an environment.** The traditional view of exploring an environment is based on the great geographic explorers of the XIX-th century: a person moving around in the environment, recording sights and sounds, taking measurements and communicating with people. Superficially this looks like an excellent application for agent mobility, as a mobile agent visits remote hosts, collecting information at every step.

The case of the exploration, however, is an anti-example, one of those cases, where our intuition about the anthropomorphic operations lead us to incorrect decisions.

The first misconception concerns the nature of communication and sensing in the physical and virtual world. Humans prefer direct sensing and communication. We prefer to "see with our own eyes", and prefer in-person meetings for important communications. Our current remote communication equipment can not convey the full input of our senses.

The situation is completely different in the virtual world.

An agent does not communicate "better" with other agents on the same host, and it does not perceive any difference between the reading of a remote or local sensor - besides the influence of bandwidth and latency. Agents can communicate their full sensor input, which is not possible for humans.

Another misunderstanding is related to the nature and cost of mobility in the physical vs. the virtual world. In the physical world mobility is one of our basic attributes. This is not true for the virtual environment. The implementation of agent mobility involves the cloning of the agent to the destination host, followed by the termination of the agent on the source host. This is a very complex operation which can fail in unexpected ways (for example, it is possible that as a result of a communication failure we end up with two running agents which claim the same identity).

For an exploration scenario, the itinerant agent is a very bad solution. There is no justification in carrying the collected data from host to host instead of sending it directly to the place where it will be collected. The best way to perform exploration is by remote querying. If we really need to send active code to the remote location, it should take the form of simultaneous cloning to all the locations which need to be explored. If the local conditions of the remote site can not accept the full agent, a subset of the agent might be sent (splitting).

We presented these three scenarios to justify the consideration of non-anthropomorphic lifecycle operations of agents. As we have shown, the NALC operations do have analogies in the real world, although not in the lifecycle of humans. Some of these analogies seem quite natural. Our third, anti-example, shows however, that the existence of an analogy does not mean that solutions can be efficiently transferred into the agent domain.

## Implementation and software engineering aspects

Implementing the non-anthropomorphic life-cycle operations, at least at the prototype level, does not present a major challenge.

However, there is no current agent framework in which the NALC operations would be supported in a consistent manner. In the following we provide the current status of the implementation of such a framework in the Bond agent framework.

### Implementing non-anthropomorphic operations in Bond agent framework

The Bond agent system (Bond 2003) is a FIPA compliant agent development environment. As the high level architecture (Figure 1) shows, the Bond system integrates a number of high quality open source tools. The communication framework and the strategy model is built on top of Java Agent Development Environment (JADE) framework (Jade agent system 2004). The knowledgebase model is using the Protégé-2000 (Grosso *et al.* 1999) ontology editor. The scripting support is based on the Jython (Jython

scripting language 2003) implementation of the Python language. Reasoning is implemented using the Jess expert system shell. Special effort was put into creating an intuitive, user friendly system (Figure 2). The Bond platform is a direct implementation of the  $AM_1^{mp}$  agent model of multi-plane state machine of active objects. The Bond system provides a systematic support for runtime mutability through an operation called *agent surgery*. Both the creation and modification of agents is relying on the Blueprint agent specification language.

The mutation abilities of the Bond agent system allow us to implement NALC operations in a relatively straightforward way. We will demonstrate this through the example of splitting and merging. We make the assumption that the splitting and merging occur on the level of individual planes.

For the *splitting* operation, we need to specify the planes which will go to the two successor agents. Usually one of the agents will maintain the identity of the previous agent. To perform the split operation, the agent is brought to a "soft stop", the currently executed actions are terminated, but new actions are not started. The internal multiplane state machine structure of the agent is read and two blueprints generated. The first one creates a new agent, and creates a set of planes in it which are copies of the corresponding planes on the original agent. The second blueprint removes some or all the planes of the original agent which are duplicated in the new agent. Some of the planes might be present in both successor agents, but all the planes of the original agent should be contained in at least one of the successor agents. As the final step the knowledgebase of the agent is copied in the new agent and both agents restarted. Potentially, a garbage collection operation can remove the parts of the knowledgebase in the two agents which are not relevant any more to their current planes. In addition to the resource management benefits, this step makes future merging operations significantly easier.

For the *merge* operation, the two agents are brought to a soft stop and the planes of one agent copied into the other agent through the help of a blueprint script. If there are duplicated planes in the two agents, they can be collapsed in the same plane, provided the knowledge associated with the planes is identical. Then, the two global knowledgebases need to be reconciled into a single knowledgebase, and the resulting new agent restarted.

The major theoretical problems for these operations are the reconciliation of the knowledgebases, potentially conflicting goals and the problem of identity of agents (Bölöni 2004).

### The software engineering aspects of NALC operations

The presence of non-anthropomorphic operations in an agent's lifecycle requires adaptations of the software engineering approaches.

At the *analysis* phase, we need to identify the potential benefits the NALC's might bring to the problem at hand. This might involve rethinking the problem specification using different language. As we have shown in our anti-example the formulation of the problem in terms of "explo-

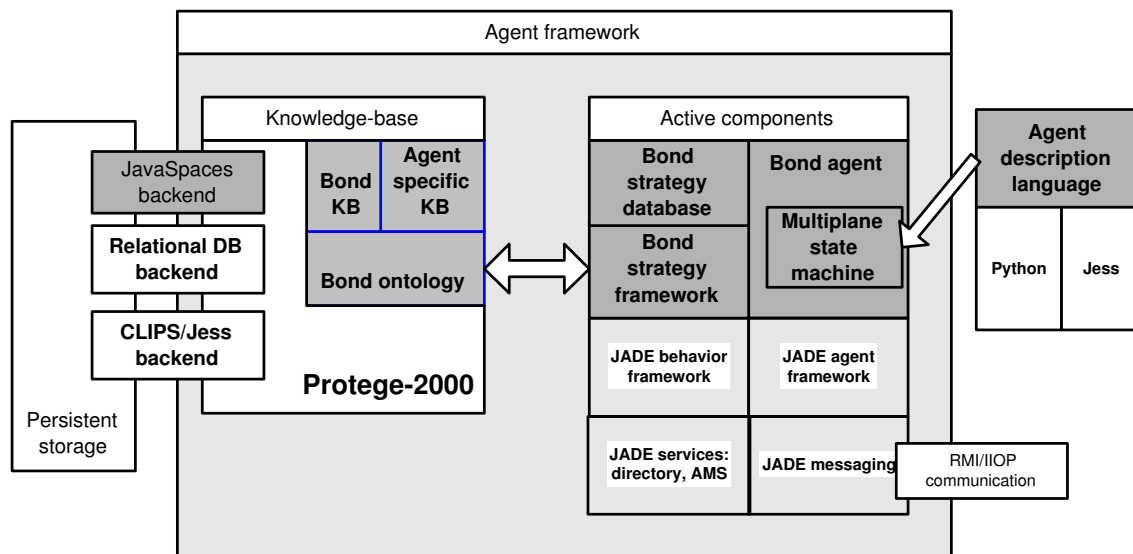


Figure 1: The high level design of the Bond agent system

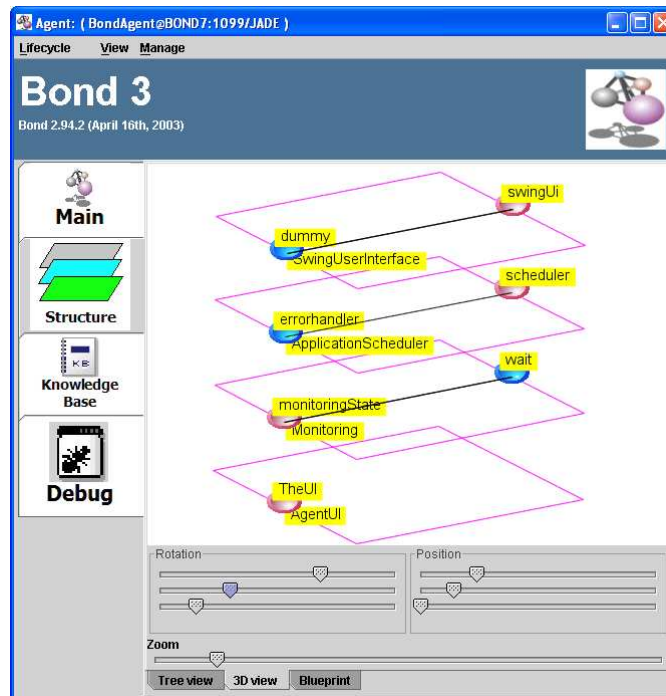


Figure 2: A screenshot of the Bond agent system

ration" leads to the incorrect model of the solitary migrating agent. A different wording, such as "data collection" might lead us to consider a wider range of solutions.

At the *design* phase, we need to implement the agents using frameworks which support the non-anthropomorphic operations planned. While in some situations it might be possible to plan these operations in advance, in most cases we need to specify *triggers* which, depending of the current

status of the agent and the perception of the environment are starting the non-anthropomorphic operations.

A modified version of the the GAIA (Wooldridge, Jennings, & Kinny 2000) agent design methodology which allows for the use of mutable agents and several varieties of NALC operations is presented in (Bölöni *et al.* 2004). We are currently working on the requirements of Agent UML (Odell, Parunak, & Bauer 2000) methodologies for NALCs.

## Conclusion

In this paper we have explored the topic of agents with non-anthropomorphic lifecycles. We argued that while biological analogies are a powerful tool, concentrating exclusively on operations for which immediate analogies can be found can lead us to overlook approaches which might be better fitted for the virtual world in which software agents live.

We have seen that the  $AM_1^{mp}$  model employed by the Bond system can be relatively easily modified to model anthropomorphic operations. Our intuition tells us that models of agency based on modal logic such as BDI (Rao & Georgeff 1999) can also be adapted for the non-anthropomorphic operations such as merging or splitting. The concrete details of such an adaptation are subject to our future research.

## References

- Bölöni, L.; Khan, M. A.; Bai, X.; Wang, G.; Ji, Y.; and Marinescu, D. C. 2004. Software engineering challenges for mutable agent systems. Springer-Verlag, Berlin Germany.
- Bölöni, L. 2004. Software engineering challenges for mutable agent systems. Proceedings of the 17th European Conference on Cybernetics and System Research.
- Bond. 2003. Webpage. URL <http://bond.cs.ucf.edu>.
- Finin, T.; Fritzon, R.; McKay, D.; and McEntire, R. 1994. KQML – A language and protocol for knowledge and information exchange. In *Proceedings of the 13th International Workshop on Distributed Artificial Intelligence*, 126–136.
- Foundation for Intelligent Physical Agents. 1997. FIPA 97 specification part 2: Agent communication language. Version 2.0.
- Grosso, W. E.; Eriksson, H.; Ferguson, R. W.; Gennari, J. H.; Tu, S. W.; and Musen, M. A. 1999. Knowledge modeling at the millennium (the design and evolution of Protege-2000). Technical report, Stanford Medical Informatics Institute.
- Jade agent system. 2004. Webpage. URL <http://sharon.cselt.it/projects/jade/>.
- Jython scripting language. 2003. Webpage. URL <http://www.jython.org>.
- Odell, J.; Parunak, H.; and Bauer, B. 2000. Extending uml for agents. In Wagner, G.; Lesperance, Y.; and Yu, E., eds., *Agent-Oriented Information Systems Workshop at the 17th National conference on Artificial Intelligence*, 3–17.
- Rao, A. S., and Georgeff, M. P. 1999. Modeling rational agents within a BDI-architecture. In *Proceedings of Knowledge Representation and Reasoning (KR&R-91)*, 473–484.
- Shehory, O.; Sycara, K.; Chalasani, P.; and Jha, S. 1998. Agent cloning: An approach to agent mobility and resource allocation. *IEEE Communications* 36(7):58–67.
- Wooldridge, M.; Jennings, N. R.; and Kinny, D. 2000. The Gaia methodology for agent-oriented analysis and design.