# An Agent Pattern Language for a More Expressive Approach

Nuno Meira      Ivo Conde e Silva      Alberto Silva

napfm@camoes.rnl.ist.utl.pt      ijcs@camoes.rnl.ist.utl.pt      Alberto.Silva@inesc.pt

INESC & IST (Technical University of Lisbon)
Rua Alves Redol, nº9,1000 Lisboa Portugal

**Abstract.** We present a set of agent patterns, that reflect a tighter connection with the real world than previously documented agent pattern languages. In the real world there are many pattern behaviors, good metaphors, that if transported to agent based applications will increase not only its capabilities and potential, as well as, possessing more expressiveness since they translate familiar concepts, make this emerging technology more easily accepted by the programming community.
In particular, we propose the following agent patterns: Receptionist, Secretary, Session, Mobile Session, Antenna, Private Session and Meeting with Moderator. We give an overview of their relationships and show, through a simple but effective example, their concrete applicability.

## 1 Introduction

The use of patterns has been proposed by other authors [2,11], as essential tools for intelligent software engineering. We agree, and wish to enhance agent technology with our perspective about Agent Patterns.

An agent is a software entity with well defined identity, state and behavior and represents, in some way, its owner in the accomplishment of tasks. The definition of agent here presented is by no means unique as there are many other attempts to define an agent. Being the issue of defining an agent outside this paper scope, we felt that the one we present is sufficiently clear and generic to serve the main purpose of this paper. A good compilation of agent definitions can be found in [10].

After some research on Agent Patterns [2,4,6,7,12,13,15], we found many patterns that described goodsolutions but lacked some expressiveness we feel is essential for agent-based application development. The real world provides us with many good metaphors and analogies for raising the expressivness of such patterns. However, many times a pattern by itself takes a strange shape and it is hard to find a good metaphor or establish a good analogy, which means that, from our point of view, it is preferable to find an entity that contains a set of pattern behaviors (including the pattern in issue), that has a well defined form and from which may be possible to establish a direct relationship with its behavior.

Some agent patterns that we are going to mention are based on patterns already defined by other authors and presented on papers to which we will make proper reference. However the work around this patterns of giving them the referred expressiveness recurring to metaphors, analogies and entities to which that type of behavior can be delegated to. Other patterns will be new and constitute behaviors and

solutions we got directly from the real world. These patterns catch solutions to problems like: how do agents communicate with other agents, located at the same place, without knowing which agents there are and how to reach them; how to contact or maintain contact with moving agents and others. Therefore, we think these patterns will capture your interest and bring you closer to this technology.

In the next section, we will briefly describe the pattern template that will be used for defining the agent patterns presented on Section 3 of this paper, namely the Receptionist Pattern, the Secretary Pattern, the Session Pattern, the Mobile Session Pattern, the Antenna Pattern, the Private Session Pattern and the Meeting with Moderator Pattern.

In Section 4 "Relating the Patterns", we give an overview and justification of each pattern's existence by relating it to the other patterns to better discuss their differences.

In order to illustrate the potential of the defined patterns, we describe on Section 5 a brief and simple example where the potentials of the proposed patterns are used together. Finally, on Section 6, will conclude this paper and introduce future work where the presented patterns have an important role.

## 2 Pattern Template

We present in this section the pattern template used in the main section of this paper. Basically, our pattern template is an adaptation of the minimum format proposed by D.Deugo [2]. The template contains the headings that follow.

**Introduction** – Introductory description to the pattern recurring to metaphors and/or analogies.

**Name** – Identification of the pattern using a representative name.

**Problem** – Definition of the problem to be solved by the pattern

**Context** – Description of one or more situations in which the pattern is applicable.

**Forces** – Description of the factors that may influence the decision of when should the pattern be applied

**Solution** – Description of the solution generated by the pattern

**References** – Indication of bibliographic references and/or patterns that inspired the defined pattern, in case they exist.

# 3 Patterns

## 3.1 Receptionist Pattern

**Introduction:** Usually a person knows from start what is to be done in a given place although many times she has to obtain information when it gets there, like knowing where to go or whom to talk to in order to get information or obtain existing services. Sometimes you do not even know what kind of services there are so you have to ask. This kind of behavior can be seen as a recurring situation on our daily life. It can and usually has a simple solution which is to have a Receptionist that provides to others this kind of information. It can also be used to one register oneself at the entrance, to ease certain kinds of interaction like warning someone that it is time to leave, or that there is someone who wants to talk with her. This solution can be easily transposed and adapted to the agent software paradigm.

**Name:** Receptionist

**Problem:** How do agents communicate with other agents, located at the same place, without knowing which agents there are and how to reach them?

**Context:** A multi-agent system is being developed and collaboration between agents is necessary. In order to accomplish tasks, agents have to communicate with other agents physically located on the same machine.

**Forces:**

- When an agent reaches some place, it may not know what kinds of services are available at that place

- An agent knows which services are available, but does not know hot to contact the agents that provide those services

**Solution:** Use a "Receptionist" static agent, that contains: a general description of the existing services, the contact of every agent that provides those services as well as a description of what each agent does and finally a list of contacts of remaining agents present on that location. An agent, when reaching a place, registers itself at the Receptionist and asks what services are provided, if there is an agent that can provide a specific service and its respective contact. For this kind of interaction, the use of KQML [5] should be appropriate since this information exchange protocol is highly adequate to this kind of situation.

**References:** Inspired by the Facilitator pattern defined by Yariv Aridov and Danny B.Lange [7], but where the name 'Receptionist' is associated with a well-known entity. An entity we can easily associate not only with the behavior provided by the facilitator but also with other pattern behaviors, therefore not adding just one more strange entity for which there is no direct association with the real world.

## 3.2 Secretary Pattern

**Introduction:** In different kinds of work places there are secretaries that help their bosses accomplish their tasks like reminding them of their agenda, monitoring their tasks, present them intermediate results, contact people working for him and so on.

**Name:** Secretary.

**Problem:** How to contact or maintain contact with moving agents?

**Context:** You are developing an application incorporating mobile agents. In order to accomplish tasks, agents have to reach others mobile agents that keep changing places, and therefore, their present locations are unknown.

**Forces:**

- Mobile agents change their locations frequently

- An agent needs to contact another agent but does not know its present location

- An agent's owner wants to contact it to terminate a task or receive intermediate results, but does not know its present location.

**Solution:** By using the Secretary  pattern that is it self an static agent where:

- Agents register themselves to the secretary on creation.

- An agent informs the secretary everytime it changes its location, or from time to time, due to message traffic overloading issues that can arise when agents change their locations too often.

- Every time an agent wants to send a message to another agent and does not know its present location, it sends the message to the secretary instead, and the secretary takes care of forwarding the message to the destination agent if possible, or can keep them so that the target agent can request them later (like a mail box).

**References:** Inspired by the proxy agent defined by D.Deugo, F.Oppacher, J.Kerester, I.Von Otte [2], but once again we find that this pattern has no well-defined identity. That's not the case with the secretary that can be easily associated to the kind of behavior presented by the Proxy Agent, and including as well other pattern behaviors, such as the control level  by the user on the agent.

### 3.3 <u>Session Pattern</u>

**Introduction:** Usually when we initiate a conversation with someone in a given place, we stay on that place talking until the conversation ends and only then we leave. If the contents of a given conversation are important its not usually good to jump from one place to another nor do other things at the same time, because this things can interfere negatively with the conversation taking place. This kind of behavior can be seen as a simple pattern behavior at the communication level of our daily lives, and can be seen as a "session" we are having with someone with a beginning, a middle and an end.

**Name:** Session

**Problem:** How do mobile agents or services can maintain a complex conversation between themselves in an simple and efficient way?

**Context:** A mobile agent system is being developed in which mobile agents interact with either mobile or static agents in complex conversations occurring over a period of time. The typical scenario is that two agents agree to communicate, interact by exchanging messages and finally stop their conversation.

**Forces:**

- Mobile agents change their position frequently.

- Messages of different conversations arrive interleaved in a random order. Complex interactions involve many messages.

- The design should be simple and easy to use. The design should restrict agents as little as possible.

- The communication should be as efficient as possible. You need to make it possible for agents to have conversations simultaneously and manage these conversations in a simple and efficient way.

**Solution:** Introduce the concept of a session. A session is an open communication link between two agents that is represented by a session object in each agent. Examples of this type of communication include message passing, RMI or sockets. The use of sessions allows agents to manage several concerns. First, conversations are separate. Agents receive messages from a specific session object and can assign a handler to each. In this handler, the current state of the conversation can be encapsulated. Moreover, an open session indicates an ongoing interaction with a corresponding agent. Second, it is possible to separate session management from the mobility problem, We can force an agent not to change its location in a session or that such a change implicitly closes all open sessions. This allows a direct and simple communication between agents, which results in less overhead and higher performance.

**References:** This is the Communication Session defined by D.Deugo , F.Oppacher, J.Kerester, I.Von Otte [2]. We felt this should be here included since it introduces other patterns described later in this paper, namely the Mobile Session and the Private Session.

### 3.4 Mobile Session Pattern

**Introduction: Often**, in order to talk to someone we do not need to be or stay on the same given place. For instance, when someone talks to another using a mobile phone she does not need to stop going where she's going.

This kind of behavior can also be seen as a pattern behavior on our daily life, and therefore be seen as a session we are having with someone that has a beginning, middle and end.

**Name:** Mobile Session

**Problem:** How do agents communicate with each other without having to worry about the mobility problem?

**Context:** A mobile agent system is being developed in which mobile agents interact with either mobile or static agents in complex conversations occurring over a period of time. The typical scenario is that two agents agree to communicate, interact by exchanging messages and finally stop their conversation, but where in the middle of this process one or both agents have changed locations.

**Forces:**

- Mobile agents change their position frequently.

- Agents need to maintain a communication session and be able to change their locations at the same time.

- The design should be simple and easy to use.

- The design should restrict agents as little as possible.

- The communication should be as efficient as possible.

- The fact that agents change their locations should not increase the session's complexity.

**Solution:** Introduce the concept of Mobile Session and Antenna (another pattern presented later in this paper). The mobile session is a communication link between two agents that is represented by a mobile session object in each agent and where messages should go through the antenna which is responsible for forwarding the messages.

The use of mobile sessions allows agents to solve several concerns related to the need they have to move. The conversations are separate. Agents receive messages from a specific mobile session object and can assign an handler to each, like in the session concept, and the details involved with their mobility can be encapsulated and solved by the antenna.

**References:** Inspired by the Direct Coupling pattern defined by D.Deugo, F.Oppacher, J.Kerester, I.Von Otte [2] but adapted to a solution we feel is closer to the real world, without introducing any strange entities and providing a greater level of abstraction, since the agent never needs to know its communication partner location.

### 3.5 Antenna Pattern

**Introduction:** On the mobile session pattern we talked about mobile phones and, as we know, this phones use an antenna and that antenna is responsible for keeping the connection. When a phone starts getting out of range the connection enters the scope of another antenna so that the call does not break, and this happens without the participants knowledge.

**Name:** Antenna

**Problem:** How do agents communicate with each other without having to worry about the mobility problem?

**Context:** You want to support the mobile session concept already introduced in this paper and therefore the context is a mobile agent system where agents can take advantage of the simplicity of use of a mobile session.

**Forces:**

- There is a need to keep the mobile session concept simple and separate from the way the message destination problem is handled, given the agents mobility.

- The fact that the agent maybe having multiple sessions with different agents and be moving at the same time should not increase the complexity of the mobile session

- The solution should consider the message traffic problems that this technology introduces and provide a way of control in the system.

**Solution:** Use an Antenna static agent. Each location (or at least each agent server) should have an agent of this kind that will be responsible for maintaining mobile sessions occurring between agents at a given place (by launching tasks for each link). Two agents communicating at the same location will be using the same antenna. When one of the agents moves, the antenna will be informed of that movement, using then the antenna that resides on the moving agent's destination. Note that even if an agent moves to more than one location he can always make sure that the number of antennas is reduced to a minimum, that is, two at the maximum (for only two agents). Also note that when an agent moves he only needs to tell the antenna and not all the agents with which the agent is having mobile sessions.

**References:** See the Mobile Session Pattern described on Section 3.4.

### 3.6 Private Session Pattern

**Introduction:** When you talk to someone and have a need for confidentiality you usually setup a safe place for that conversation to take place, and both intervinients move there and then start the conversation. This kind of behavior can be seen as a pattern behavior at the security level on our daily lives communications, and can also be seen as a private session we are having with someone that needs some restrictions for security reasons.

**Name:** Private Session

**Problem:** How do agents communicate with each other in a safe way?

**Context:** A mobile agent system is being developed in which mobile agents interact with either mobile or static agents in complex conversations occurring over a period of time. The typical scenario is that two agents agree to communicate, interact by exchanging messages and finally stop their conversation, but where some specified security restriction should be considered and the conversation must occur in a safe place.

**Forces:**

- Mobile agents and other agents have to engage communication on a location that obeys security restrictions.

- It is necessary to agree on the location where the communication will take place.

**Solution:** Introduce the concept of Private Session. A private session is a kind of session, represented by the private session object, where previous to the conversation itself, negotiation takes place in order to define the location of the session. The private session object should have a list of safe locations for the session to take place, or any other way to determine it.

**References:** See also the Session Pattern and the Mobile Session Pattern referred earlier in this paper.

### 3.7 <u>Meeting with Moderator Pattern</u>

**Introduction:** Often we need to discuss a given subject with more than one person we usually setup a meeting, hope everyone appears and then start the conversation without having the need to repeat the same thing to each of the people involved. In such a meeting there is normally a person whose job is to coordinate the meeting which we call a moderator. A clear example of this could be a Parliament. This is also a recurring situation on our daily lives.

**Name:** Meeting with moderator

**Problem:** How can an agent communicate over a given subject with more that an agent in a simple way?

**Context:** A multi-agent system is being developed and collaboration between agents is necessary. In order to accomplish tasks, agents must communicate dynamically with other agents.

**Forces:**

- Agents need to maintain communication with other agents over a common subject

- An agent does not need to worry constantly about sending messages to multiple agents.

- It is useful to abstract the synchronization in time and place needed for the interaction to take place.

**Solution:** Using a 'Meeting with Moderator' agent. Every time there is a need for a conversation between several agents, a 'Meeting with moderator' agent is invoked. This agent is responsible for initiating a session (that can be either a session, a mobile session or even private session) well suited for the situation, synchronizing messages between agents and terminating communication sessions.

**References:** Inspired by the Meeting pattern defined by Yariv Aridov, Danny B.Lange [7], but where there is no restriction to local interaction.

# 4 Relating the Patterns

Relating the patterns is probably the best way to justify their existence, through their conceptual distinctions and also clarifying the needs for such distinctions.

Starting with the Receptionist and Secretary, we can ask ourselves why shouldn't they be the same agent?. The answer in this case is a simple one. The receptionist role is an informative one, that is, it should not have and does not have any kind of control over the agents.

On the other hand, the secretary's role has to do a lot with agent control, that is, doing tasks in order to supervise, facilitate or coordinate the agent's jobs. As such, it also makes sense to exist a secretary for a given group of agents (or even several), while there is no need to have more that one receptionist per place. We can see the secretary as the skeleton of a kind of service agent that can take a distinct shape, accordingly with the kind of work the agent that works for the secretary, its capacity to manipulate that work's results and the way it can be presented to his creator (a user or any other agent owner). Furthermore, nothing forbids an agent that works for a secretary to be also a secretary. We think the reasons just presented are more than enough to justify the distinction, and certainly would not be difficult to find even more.

A similar question could be posed of the Antenna pattern: why shouldn't the Receptionist or the Secretary do what the Antenna is doing? The answer to this question may not be as straightforward as the previous one, but still exists and is conceptually important. As far as the secretary is concerned there are two important issues to consider.
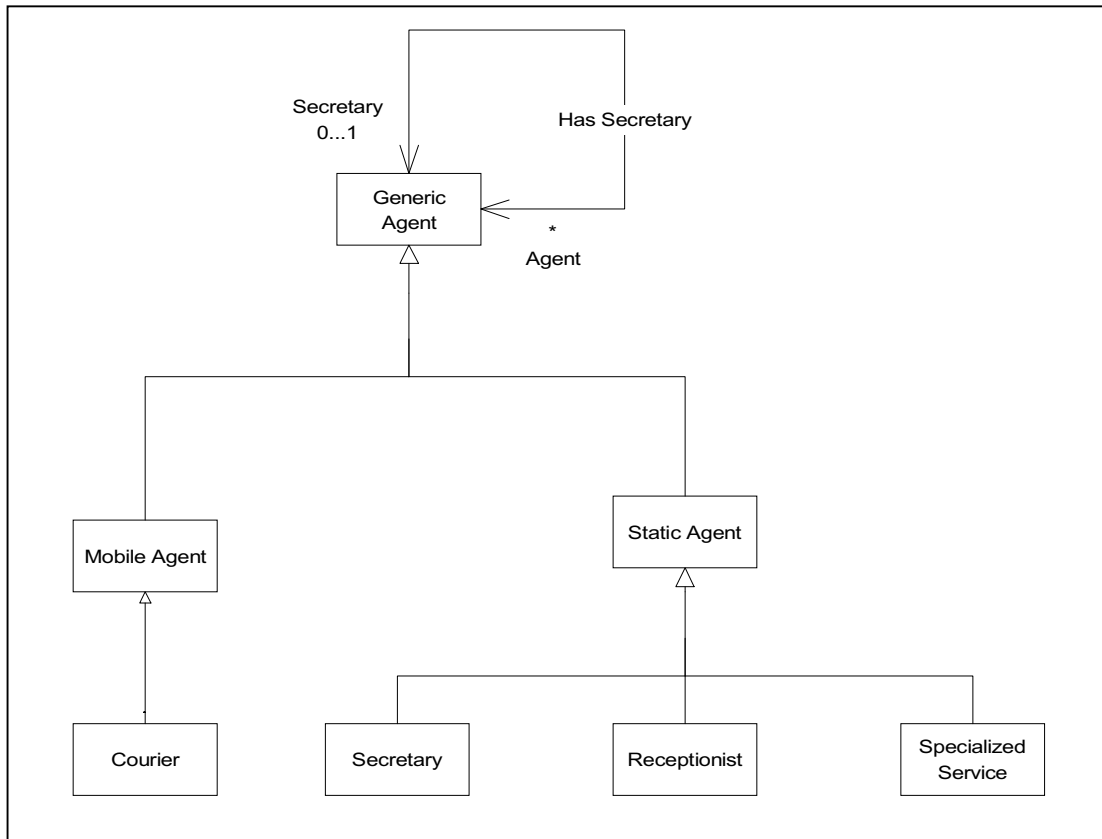
First, an agent does not necessarily need to have a secretary, which would mean in this case that without the antenna functionality the agent would be impeached to have a mobile session. Second, if the Secretary acts like an Antenna, and knowing the fact that it resides on the user's machine, then we could not take advantage of what distributed computing has to offer to the agent technology.

In what concerns the Receptionist, it does not make sense in conceptual terms to have it being responsible for the communication process between agents (or even being capable of). That's the antenna responsibility. The Antenna represents a structure, like a building or a car, and as such it makes sense that it belongs to the system structure, i.e., as a component of the agent support system. Furthermore, only this way can the system administrator have some control over the extra message traffic originated by this technology, a problem for which in the future should be necessary to provide easy solutions.

In what concerns the remaining patterns (Session, Mobile Session, Private Session, Meeting with Moderator) we feel there is no need to face them against each other, as this patterns only represent four distinct ways of well known interactions and therefore we think no special issues are raised here.
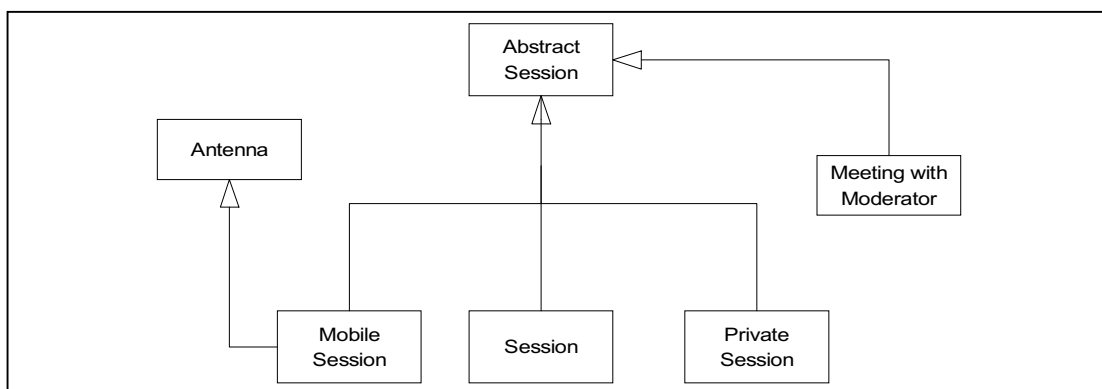
In order to further illustrate how this patterns relate and to map them in a agent hierarchy we present the diagram that follows. The diagram (a UML[9] diagram) also shows some of the most common relationships of use between agent classes.

We start by the general agent class and further subdivide it with concern for an important agent property: mobility. Some of the leafs on the tree represent the agents that correspond to the solutions generated by the patterns presented on this paper, namely the receptionist and the secretary. The specialized service agent corresponds to a static agent that provides a domain specific service in a given place. The courier agent can be seen as the 'common' mobile agent that carries information from one place to another.

**Agent Hierarchy**

The following diagram represents the hierarchy between the remaining patterns defined on this paper. We hope the example presented on the next section can further elucidate some of the relationships presented on these two UML class diagrams.

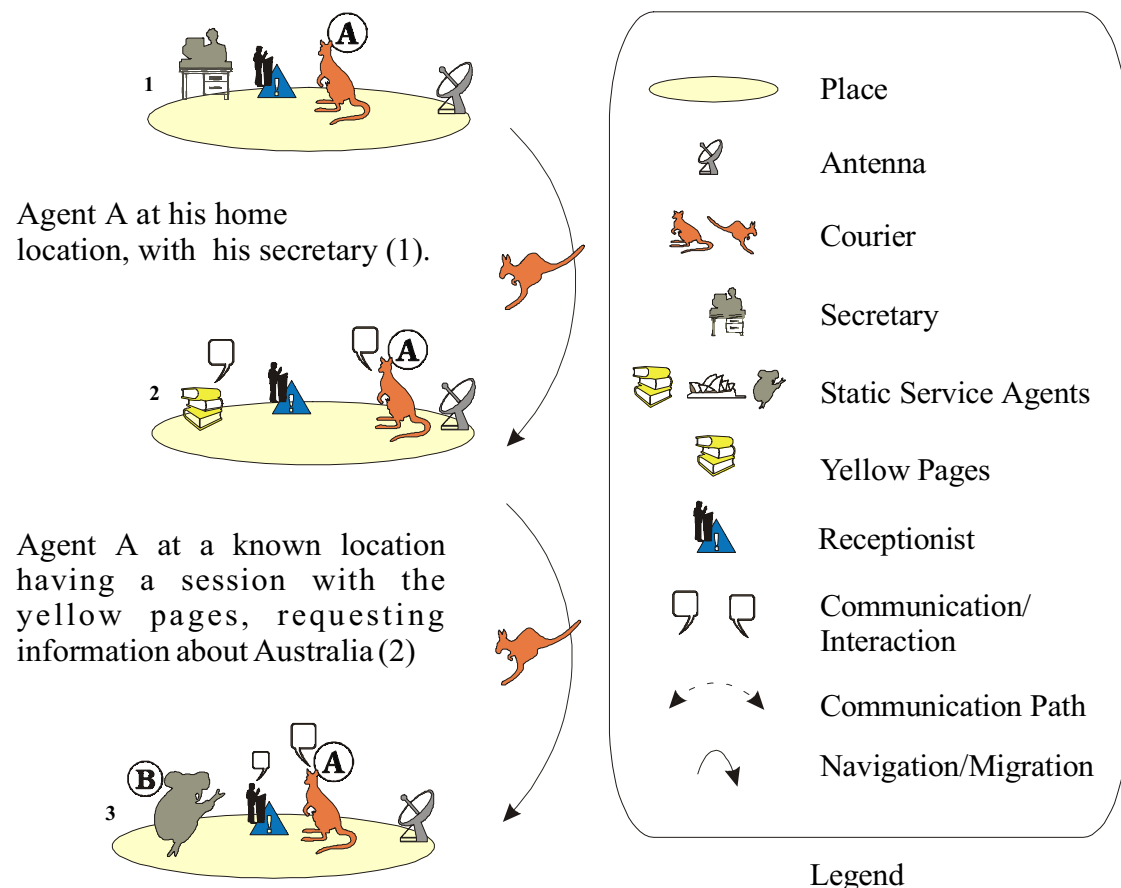**Communication Mechanisms**

## 5 Applying the Agent Pattern Language: An Example

To conclude, we will present an example where the potentials of the presented patterns are used together, and how these can be related to each other.
Let's consider the following context:

- There is a Courier (mobile agent) and his goal is to get information about Australia.
- The agent has a secretary
- The agent only knows a static service agent, which we will conveniently call 'yellow pages'.
- The static service agent ('yellow pages'), provide services and can be seen as a search engine. This agent, has information about what kind of services an agent is able to get in a certain place, but does not have the contacts of the agents which provide that services.

Agent A at his home
location, with his secretary (1).

Agent A at a known location
having a session with the
yellow pages, requesting
information about Australia (2)

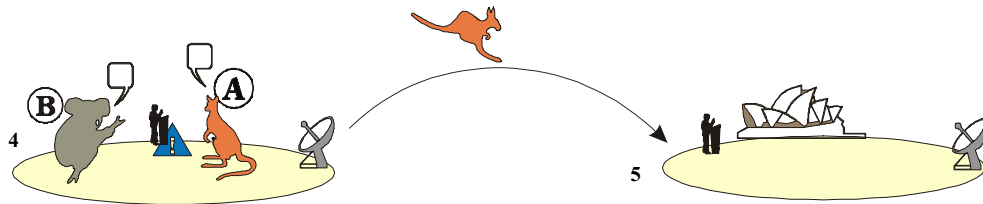| | |
|---|---|
| Place | |
| Antenna | |
| Courier | |
| Secretary | |
| Static Service Agents | |
| Yellow Pages | |
| Receptionist | |
| Communication/ Interaction | |
| Communication Path | |
| Navigation/Migration | |

Legend

Agent A at one of the locations the Yellow Pages sent him.
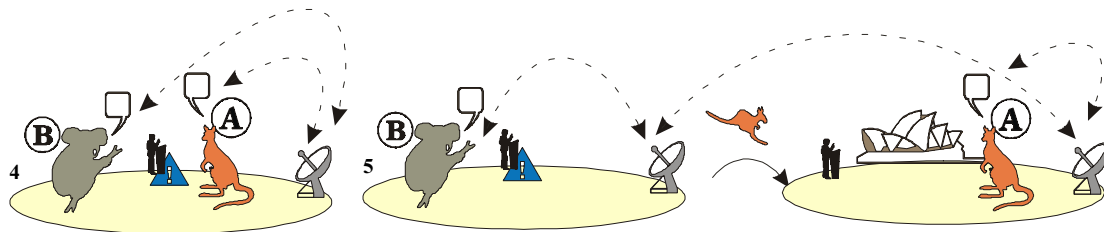Due to this place has a Receptionist, Agent A is able to request Agent B's contact (3).

After Agent A knows Agent B's contact, if they have the option to establish a Session or Mobile Session, they can communicate at least through 4 different ways.
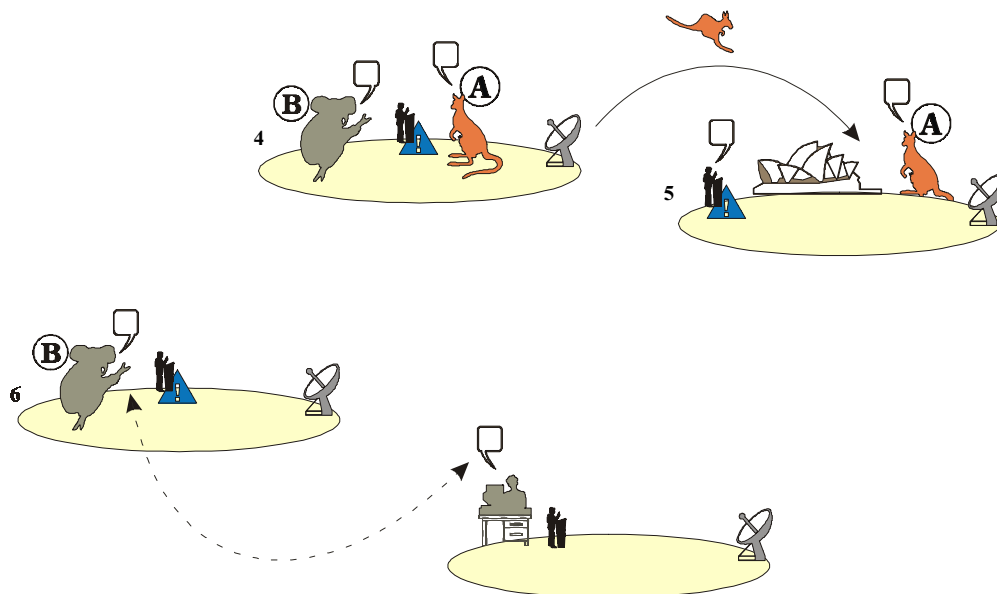
## Case 1: Communication through a Session



Agent A can start a session with Agent B, wait and get his result (1), and then go to another place (5).

## Case 2: Communication through a Mobile Session



Agent A can start a mobile session with Agent B (4) and consequently, doesn't have to wait or terminate his session, if he wants to go to another place. Due to the establishment of the Mobile Session, both agents would keep their interaction even though they are in different places (5).
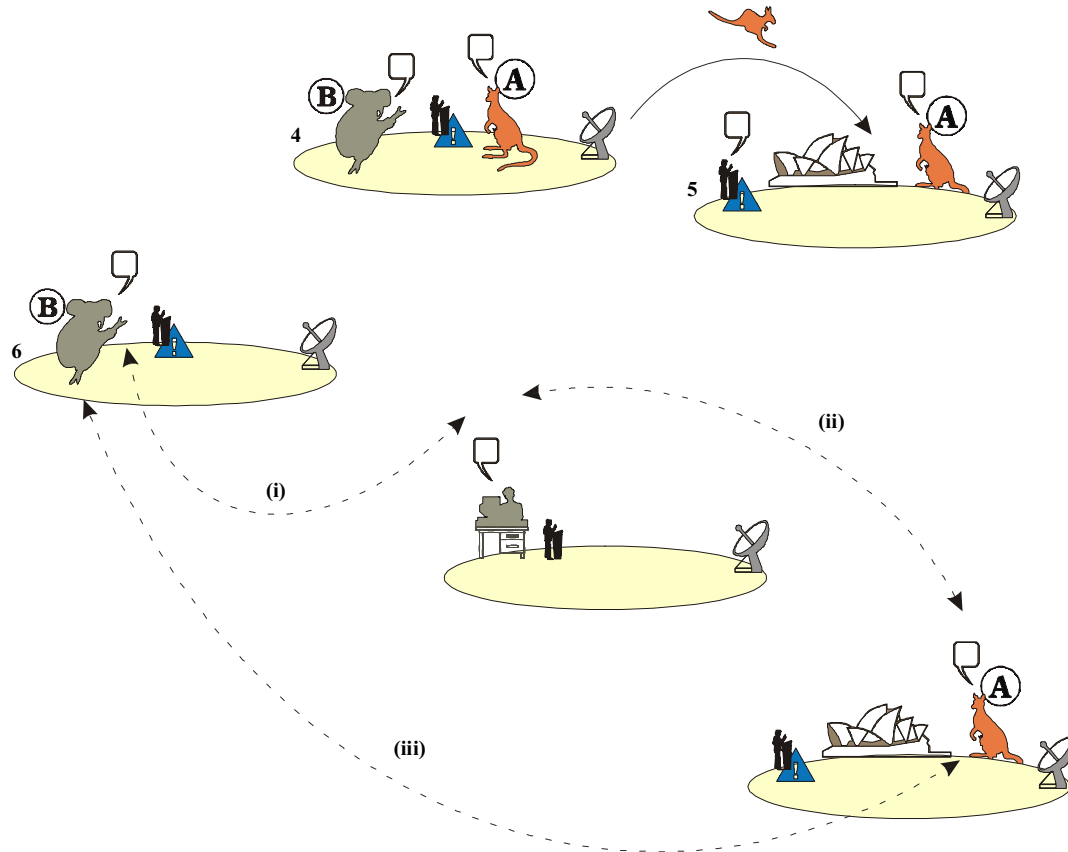
## Case 3: Communication through Agent's Secretary



Because Agent A has a Secretary, he can ask Agent B to send Agent A's secretary the results (4), and go to the next place to request more services (5). Then, Agent B sends

them to the secretary, where she gets and holds the results (6), until Agent A gets home or requests them in somehow.

**Case 4: Notification through Agent's Secretary**

Agent A can also ask Agent B to notify Agent A's secretary when the service is complete (4) and go immediately to the next place to request more services (5). Then Agent B, when his task is completed (6), notifies the secretary (i) and she gets in touch with Agent A (ii). Agent A can start a session with Agent B (iii).

There can also be agents that move from place to place, consulting local receptionists for services provided and therefore updating the yellow pages in an easy and efficient way. The reason why the yellow pages do not also keep the agents contacts, is because, the agents which provide services in a certain place can change as well as their contacts. For instance, if you want to by a computer, you only need to know where you can by it, not the person who is going to sell it.

As this example, there are many others that can be easily imagined and pictured. However, this simple example allows us to show the application and use of the proposed agent pattern language in an attractable and understandable way.

# 6 Conclusions and Future Work

Associating agent systems and concepts in a more expressive way to a perspective based on our society, will bring, from our point of view, not only a larger acceptance from the programmers community but also from future agent users. Our society is full of behaviors and metaphors that are familiar to everyone and that reflect many years of experience and therefore these patterns can generate good solutions to build agent societies.

Although it is not part of this paper main intent, we think a note should be added about the notation used on the figure of the previous section. The "cartoonish" style of the example on Section 5 is a symptom of the lack of a standard notation to represent concepts like places, static and mobile agents, agents in general, communication and interaction, messaging, mobility, and so on. Our first suggestion for an approach to this problem would be the use of a modeling language such as the Unified Modeling Language (UML) and its stereotypes [9,16]. Whatever the solution will be, future work on this subject should be important to avoid the proliferation of multiple notations, providing the community with a common language on graphical representations of agent systems, situations and others.

We would also like to refer the fact that this pattern language will not end with this paper, since they will be developed and implemented on a developing platform on top of the AgentSpace framework [1,12]. In short, this work will concentrate on two main areas.

First, development of an API based on the implementation of agent patterns (where the patterns presented on this paper are obviously included).

Second, providing a high level development environment through the use of a modeling language like the Unified Modeling Language [9], so that modelers, software engineers and programmers in general can design and develop their agent based applications in a high level and powerful way.

On a more theoretical basis, aspects of agent theory such as mobility, communication, interaction, tasks and others, are intended to be explored.

## Acknowledgments

## References

[1]     "An Overview of AgentSpace: A Next-Generation Mobile Agent System" – Alberto Silva, Miguel Mira da Silva and José Delgado – In Proceedings of the Mobile Agents'98.

[2]     "Patterns as a Means for Intelligent Software Engineering" - D.Deugo, F.Oppacher, J.Kerester, I.Von Otte - IC-AI99,502SA, 1999.

[3]     "A case for Mobile Agent Patterns" – Dwight Deugo, Michael Weiss – Carleton University, 1999.

[4]     "Communication as a Means to Differentiate Objects, Components and Agents" – Dwight Deugo, Franz Oppacher, Michael Weiss" – Carleton University, 1999.

[5]     "KQML - A language and Information Exchange" – Tim Finnin, Rich Fritzon – University of Maryland, Technical Report CS-94-02,1994.

[6]     "The Agent Pattern for Mobile Agent Systems" - Alberto Silva, José Delgado, 1998 – in European Conference on Pattern Languages of Programming and Computing, EuroPlop'98

[7]     "Agent Design Patterns - Elements of Agent Application Design" - Yariv Aridov, Danny B.Lange - Proceedings of the Mobile Agents'98, ACM Press, 1998.

[8]     "The Timeless Way of Building" – C.Alexander – Oxford University Press, 1977.

[9]     "The Unified Modeling Language User Guide" -  G.Booch, J.Rumbaugh, I. Jacobson – Addison Wesley, 1999.

[10]    " Is it an agent or just a program? A Taxonomy for Autonomous Agent" – S. Franklin, A. Graesser - Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages , 1996.

[11]    "Design Patterns: Elements of Reusable Object-Oriented Software" - E. Gamma, R. Helm, R. Johnson, J. Vlissides - Addison-Wesley, Reading, MA, 1995.

[12]    Agent Space Web Site – http://berlin.inesc.pt/agentspace/

[13]    "A roadmap of Agent Research and Development" – N.Jennings, K. Sycara, M.Wooldridge – Autonomous Agents and Multi-Agent Systems Journal,  1998.

[14]    "Intelligent Agents: Theory and Practice " - M. Wooldbridge, N. Jennings – Knowledge Engineering Review, 10(2), 1995.

[15]    "The Layered Agent Pattern Language" - E. Kendal et al. – Proceeding of the Conference on Pattern Languages of Programs (PLOP'97), 1997.

[16]    "Extending UML for Agents" – J. Odell, H. Parunak and B. Bauer. – submitted paper, 2000.