Adaptation in Open Systems

Reflection as a Backbone

Sylvain Giroux
IRO, Faculté des arts et sciences,
Université de Montréal,
Montréal, Canada, H3C 3J7
email: giroux@iro.umontreal.ca

Alain Senteni
IREMIA, Faculté des sciences,
Université de la Réunion
97489 St Denis, La Réunion
email: senteni@iremia.fr

Guy Lapalme
IRO, Faculté des arts et sciences,
Université de Montréal,
Montréal, Canada, H3C 3J7
email: lapalme@iro.umontreal.ca

Abstract

This paper presents the design principles behind ReActalk implementation: ReActalk uses an organizational approach of reflection and is meant to provide a framework for the study of adaptation in the context of agents and open systems. Since open systems interact with a fluid reality, they must adapt in order to mitigate the consequences of the narrowness and the brittleness of traditional systems. Accordingly, ReActalk uses reflection as a framework for adaptive processes evolving in a dynamic environment. We show, on the one hand, how reflection can implement adaptive processes, and, on the other hand, how adaptive mechanisms are actualized in ReActalk. First we define, based on Gregory Bateson's work, adaptation viewed either as a modification of one's behavior or as a modification of one's environment. Then we outline how to get from Bateson's view to an architecture suited for adaptation. Finally, we show how ReActalk allows both types of adaptation.

1 Introduction

This paper presents the ins and outs of ReActalk[1] [15], a reflective actor[2] kernel dedicated to adaptation, written on top of Smalltalk-80. Indeed object-oriented programming (OOP) provides a promising framework for the integration of heterogeneous systems where encapsulation, preservation of locality and message passing appear as essential dimensions. Multi-agent systems add concurrency to these needs [6]. However we still need an integrating perspective on all those concepts in the context of open systems (OS).

OS are by definition interacting with a fluid reality. Ineluctably they must adapt in order to overcome the narrowness and the brittleness of traditional systems [4]. Furthermore an adaptive system would have more chances to evolve on its own without breaking down when faced with unexpected situations, and therefore releasing users and designers from intervening. Since adaptation smooths the interactions, it may also make easier the integration and the cooperation of heterogeneous systems.

Moreover, if we render OS adaptive, we can take advantage of their openness and their adaptiveness in the construction of complex systems. Ultimately, instead of building systems from A to Z, trying to foresee all the harmful interactions, if parts are considered and designed as adaptive OS, we would "just" need to put them together in order to build the system: parts will cooperate and adapt to each others and the overall system will emerge from their interactions.

Both openness and adaptation seem the key concepts and consequently they must be integrated in a supporting architecture. An OS gets its openness from its embedding in larger systems. From the viewpoint of an agent, adaptation relies either on self-modification or modification of its environment. ReActalk's design puts forward both characteristics.

In the paper, an OS is viewed as an ecosystem where (re)active agents, passive resources and external influences crystallize their interactions: interpreted from outside, agents seem to cooperate, thus cooperation found expression throughout adaptive mechanisms. Adaptive mechanisms, guided by constraints internal to an agent (goals, abilities...) or external (other agents, available resources...), aim at balancing the ecosystem. Such an equilibrium is either interpreted as the solution of a given problem as in ecoproblem solving [10] or corresponds to a configuration favorable to the solution of the problem as in [13].

Therefore the need for a general framework designed to support and structure adaptive mechanisms has guided the design and implementation of ReActalk. ReActalk proposes an organizational approach to reflection in the actor paradigm and is meant to provide a framework for the study of adaptation in the context of actors and OS. While most reflective systems, such as [21], use reflection to provide hooks on the programming language and its environment, for the user to open it and customize it, our

1 ReActalk stands for reflective actors in Smalltalk-80.

2 From our standpoint, the distinction between an *agent* and an *actor* depends on the viewpoint : the term "actor" refers to an entity taken in isolation whereas an agent is an actor understood as a member of an ecosystem. We use the term *organization* to express the reification of a standpoint on an ecosystem.

0-8186-3135-X/93 $03.00

approach differs. We use reflection as the essential dimension of adaptive processes evolving in a dynamic environment.

A closer look at adaptation helps identifying the relevance of the following concepts: individual, environment and the relation between an individual and its environment. These elements form the basis of ReActalk. Starting from the need for OS to become adaptive, the main goal of this paper is, on the one hand, to show how reflection can implement adaptive processes, and, on the other hand, how adaptive mechanisms are actualized in ReActalk.

The paper is articulated around three major sections:

- Section 2 presents the principles that lead to the design of ReActalk. First of all Gregory Bateson's perspective on adaptation is put forward. Then we derive a partial ontology. Finally ReActalk is described as an instantiation of this ontology.

- Section 3 shows why and how reflection provides an adequate framework to support adaptation, viewed as a modification of one's behavior. This section is closed by the description of some of the adaptive mechanisms implemented within ReActalk. They act in such a way that heterogeneous agents can dynamically and temporarily modify their model of computation. These modifications allow them to work together.

- Section 4 follows the same path, but deals rather with adaptation, viewed as a modification of one's environment.

ReActalk is also compared to related works. Paths worth investigating are finally outlined.

2 A platform for the study of adaptation

This section begins with a discussion on the perspective adopted upon adaptation and its consequences. From this perspective, we derive at first a partial ontology, then a general architecture for the exploration of adaptation, and finally, ReActalk, as an implementation of this architecture. This section deals only with the ontogeny of ReActalk, while the next one shows how it can be used to actualize the adaptation process.

2.1 Adaptation

Although there is a huge amount of work on adaptation, we have selected Gregory Bateson's perspective [2] [3]:

> *Adaptation. A feature of an organism whereby it seemingly fits better into its environment and way of life. The process of achieving that fit.* [3, p. 249].

Bateson's approach seems general enough to be transposed into computer science as well as precise enough to be applied. Furthermore, it brings out several important topics related to OS and real world constraints. First of all, it underlines the importance of the link between an organism and its environment: from this kind of systemic perspective, an organism cannot be understood unless it is viewed as a component of a larger system. The openness of OS lies in their embedding into larger ones. Secondly, Bateson suggests two ways for an organism to adapt:

- either to self-modify, i.e. modify one's behavior;
- or to modify one's environment.

The first viewpoint suggests reflection as a framework for adaptation. The second one supposes the ability to modify its environment. Thus, adaptation is driven by the triad:

- the individual perceived as a whole by its environment;
- the environment as an organization of individuals and relations among them;
- the relation as the interaction between an individual and its environment.

Inspired by this triad, the organizational reflection embodied in ReActalk combines both methods.

2.2 Individuals, organizations and relations

This section states axioms on the nature of the world, deduces the connections between individuals, organizations and relations and uses the conclusions to define an architecture.

2.2.1 An intuitive ontology

Since computer systems are bound to operate in conjunction with the world, this one appears as both a source of inspiration and a source of constraints. Consequently, the following axioms[1] should guide the definition of an architecture for adaptive systems:

A1 the world is made of objects;

A2 systems in the world are basically open: real systems are connected to and communicate with the external world;

A3 the world is basically concurrent;

A4 an entity belongs to an environment and must be understood in relation with it;

A5 an entity needs to adapt to changing situations.

[1] The ontology proposed in [28] goes in the same direction than the intuitions embodied in these axioms.

2.2.2 A coherent architecture for the design of adaptive systems

Not to start from scratch, the axioms suggested the use of actors [17], agents [12], environment [22] and reflection [24]. Nevertheless, these tools ought to be integrated into a coherent perspective. Since the interactions among entities appear to be crucial, the notion of organization seems the one able to articulate and unify these concepts:

- an actor is an organization of actors;
- an actor is reflective and thus can reason or act upon its own behavior;
- the reflective representation of an actor is an organization of actors.
- an agent is an actor part of an organization;
- an ecosystem is an organization.

From this viewpoint, the world is a set of micro-ecosystems interacting with one another, themselves embedded into higher level ecosystems. For instance, the model of a dinner for two in a restaurant is made of two ecosystems interacting in the context of a restaurant, a "higher level" ecosystem.

2.2.3 A formal definition of an ecosystem

Formally, an ecosystem can be described by a directed labelled graph <A, R>. This graph evolves dynamically by the addition/deletion of vertices and edges. The set of vertices A contains agents. The set of directed labelled edges R define relations among the agents in the context of the ecosystem. Such a relation defines a communication path between two agents, so it is a triad <S, D, P>

where S is the source agent
D is the destination agent
P, the label, is an agent who manages the communication.

Thus, the relation <a1, a2, a3> means that agent a1 can communicate with agent a2 through the communication protocol defined by agent a3. The graph is directed because if agent a1 can send messages to agent a2, the converse is not necessarily true.

Since the exchange of informations is subject to many constraints in real life, the communication agent is responsible to render them explicit. A communication agent could model the transfer of information through smell diffusion. For instance, if the ecosystem is placed in the vacuum, it censors noise message. A communication agent could also serve as an interpreter between the agents performing message translation, type transformation on argument or adapting feedback through result modification, therefore smoothing the heterogeneity among agents. Finally Watzlawick, Beavin and Jackson [31] point out that although it is possible to transmit sequences of symbols with a perfect syntactic precision, these symbols would have no meaning unless the transmitter and the receiver have agreed on their meaning. In this sense, any information sharing presupposes a semantic convention. In that case, communication agents render explicit the semantic convention of the agents.

2.3 ReActalk

ReActalk is built on top of Actalk [7], an actor platform in Smalltalk-80. ReActalk proposes an organizational approach of reflection in an actor universe. This section presents its main features. See [15] for a full description.

2.3.1 Overview of the architecture

While the actor paradigm allows the manipulation of entities on an individual and concurrent basis, ReActalk introduces in this paradigm the concepts of reflection and organizations, proposing an organizational approach of reflection. Reflective levels are viewed as ecosystems of agents (Fig. 1). An actor is described in a causally connected way by a personal meta-actor, where a meta-actor is an ecosystem of agents evolving along the time: agents and relations can join, disappear, be replaced. Finally, this reflective actor architecture leads to a model well suited for the representation of OS, where an actor can be modified or can modify itself dynamically and thus, adapt to changes in its environment.

2.3.2 Metaphors around ReActalk

As mentioned by Winograd and Flores [32], we must take social activity as the ultimate foundation of intelligibility, and even of existence. From this perspective, an entity is indissociable of its environment and their symbiosis determines the development of both [2]. This statement is the basis of our opportunist approach: an entity reacts to its environment. The reactions are triggered by the stimuli received from the environment.

In OOP, messages act as stimuli activating objects. In ReActalk, actors are reactive message-driven entities and accordingly meta-actors reify the message-processing mechanisms of their denotations. In a biological analogy, a meta-actor would reify the actor's nervous system. In an economic one, it is a partial taylorization[1] of the message-processing system. Each component of the meta-actor

[1] Taylorism was defined by F. W. Taylor. It is a system for labor organization, execution time control and workers remunerations. It is important to stress that taylorism in ReActalk is not apply as a whole, we restrict its application to labor organization. Many works in distributed artificial intelligence is likely to complete this partial taylorization, specially with regard to execution time control and remuneration.

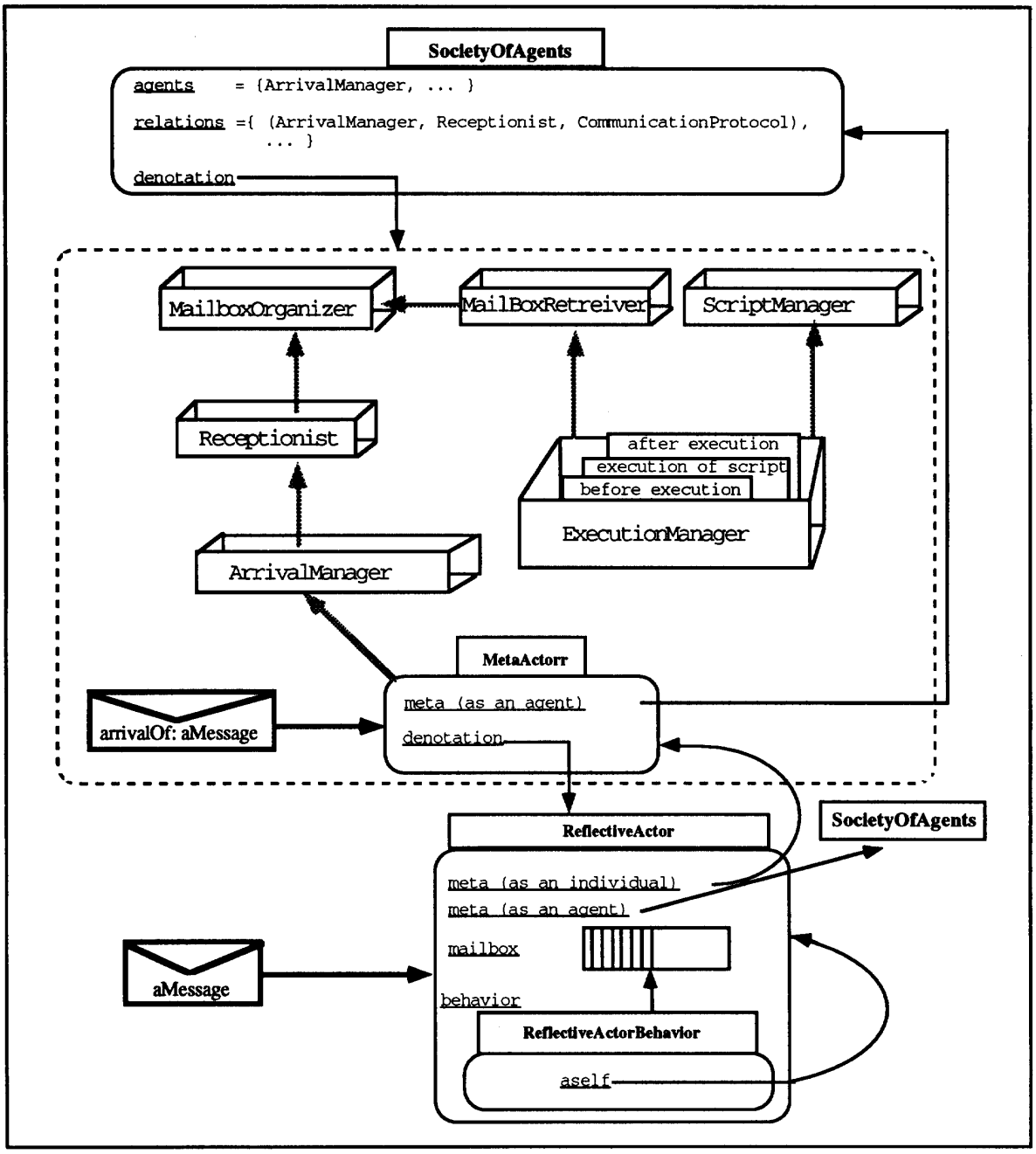

Fig. 1 An actor, its meta-actor and its meta-meta-actor. The meta-actor is a message processing factory: a message transmission to the actor is interpreted in terms of operations at the metalevel. If highlight is put on flexibility: the meta-actor is described as a society of agents by the meta-meta-actor. For the sake of simplicity, we have not shown all the metarepresentations (meta-actors, classes, ecosystems).

organization is then understood as a work station on a message-processing line, taking care of a step of message-processing. Pieces on the line are reified messages; their flow along the line embodies the relations.

This perspective upon actors is very subjective and does not pretend to be the only one, but hopefully, reflection helps integrating different perspectives. Reflection prevents from freezing a standpoint in the structure of an actor, as would a rigid taylorization of the message-processing system through instance variables such as `message-handler`, `message-receiver`...[1]

3 Adaptation through reflection

Adaptation lies on the ability to dynamically change either one's behavior or one's environment. This section puts the emphasis on the modification of one's behavior and discusses ReActalk answers to questions such as:

- When to adapt ?
- How to adapt ?
- Who is in charge of the adaptation process ?
- Should adaptation be an individual or a collective endeavor ?
- Should adaptation be permanent or temporary ?

3.1 The implementor's standpoint

The activation of the adaptation process is the first fundamental question. Several approaches are possible, from minimal to extensive. Extensive adaptation is a continuous adaptation process managed from the metalevel while minimal adaptation is a process triggered by an express request of the denotation upon failure. Implementationwise, it is a matter of concurrent versus sequential evolution of meta and denoted levels. If concurrent, the meta-actor watches continuously its denotation, gathers informations and makes decisions. If sequential, the meta-actor and its denotation should be activated alternately and adaptation is triggered by the denotation that sends its meta-actor requests for reflective operations.

The second fundamental question is : "Should adaptation be an individual or a collective endeavor ?" If collective, the adaptation process can modify:

- either the characteristics of a set of individuals (e.g. species genetic evolution);
- or the structure of relations among a group of individuals or even the individuals themselves (e.g. restructuring of a company).

3.2 When to adapt ?

Should adaptation be a continuous process or should it be activated only under special circumstances? Both scenarios are feasible and relevant for both individuals and groups. The answer dictates the parallelism between meta and denoted levels.

3.2.1 When adaptation decision making belongs to the metalevel

To conceive adaptation as a continuous process, always active, gathering data, checking for adaptation opportunities and making appropriate decisions, implies that the metalevel is in charge of all the steps. In this scenario, denoted actors need not to be "conscious" of the process. Therefore, the activities of meta and denoted levels can be concurrent.

Let's take an example in actor programming involving adaptation and efficiency:

> Let A be an actor inheriting a method M, deeply buried in the inheritance hierarchy;
> ↑A, A's meta-actor;
>
> with the focus on memory-access efficiency, if ↑A keeps an account of the frequency of method calls, it becomes easy to state method M import conditions and simulate a cache;
>
> with the focus on run-time efficiency, methods specialization according to the frequency of calls with constant arguments (e.g. through currification and partial evaluation [23]) appears as another avenue.

These incremental approaches can be completed by decremental ones, in order to maintain a "decent" number of methods (e.g. trow-away compiling [8]).

Adaptability becomes a new parameter for efficiency: run-time efficient systems unable to adapt need more revision than slower adaptive ones. A metalevel architecture seems the cleanest and the most flexible implementation, since it splits inherent behavior computation from computation related to adaptation, thus making easier dynamic changes of either of the aspects. Import conditions may as well dynamically evolve with the environment where A or ↑A perform, according to the work load of other actors or to the available processors.

The "strength" of the causal link between meta and denoted levels determines concurrency. A tight coupling prohibits any incoherence between both, while a loose one gives more room to their concurrent evolution. In its current status, ReActalk relies on a tight coupling.

---

[1] [16] shows the lack of flexibility and adaptability of this approach: the model of computation cannot evolves dynamically. The processing line structure varies according to the model of computation. Freezing its structure forbids dynamic evolution and prohibits a large range of adaptations.

3.2.2 When adaptation decision making belongs to the denoted level

What if adaptation is activated only under special circumstances, such as a fall of efficiency. In OS, this approach is particularly adequate for dealing with breakdowns although breakdown-driven adaptation shifts its recognition at the denoted level.

For instance, in case of agent overload or breaking, ReActalk's message-processing and relations reification allows to solve this kind of breakdown by re-routing the message stream towards another agent until the deficient one is again operational.

Another way of adapting to breakdown problems consists in changing the viewpoint embodied in the agent's reified representation of the external world, through a "*zoom process*", grouping or ungrouping entities and relations in order to find a vantage point:

- a *zoom out* (synthesis) results in a simplification by combining several individuals and/or relations into a whole, hiding aspects, absorbing details, but making local a global perspective on the problematic situation ;
- a *zoom in* (analysis) ungroups an organization into its components, reifying details.

ReActalk's organizational approach is fairly well suited to support this type of shifts, while self-reference and causal link make reflective operations an ideal framework for such adaptation mechanisms.

3.3 Should adaptation be an individual or a collective endeavor ?

Individual and collective aspects of adaptation are indissociable. This assumption may be illustrated by a human metaphor.

3.3.1 Analysis: from individuals to organizations

For an observer, a human being is a whole whose activities such as thinking or sleeping are performed by a single, indivisible individual. Illness (breakdown) induces an adaptation process and eventually a change in the person's internal structure or behavior. Traditional medicine provides hooks to look at individuals as "collections of organs", denoting individual selves. Self-diagnosis reifies organs and their relations and, by zooming into individuals, provide a different anatomical representation, changing individuals into "dividuals". The medical metalevel render explicit representations of anatomical aspects of the individual hidden to the world under normal circumstances and thus allows surgery to be performed. Within ReActalk, surgery consists in replacing a failing agent by a brand new one, or in defining new relations inside a meta-actor.

3.3.2 Synthesis: from organizations to individuals

Whenever a detailed anatomical analysis is not successful for establishing a diagnosis, a shift of viewpoint becomes necessary: synthesis combines organs in systems such as nervous, digestive, etc. and provides a different viewpoint, that might give the adequate perspective. The denoted level is now the groups of organs, while the meta-metalevel proceeds to a grouping according to specific criteria. There is then a shift at the metalevel from n denoted entities to m reifying ones, with $m < n$. From then on, a given organic system is considered as a whole. In this case a diet change could appear as an alternative to surgery. ReActalk would translate this by a new `mailBoxManager` rejecting any message/stimulus forbidden by the prescribed diet.

3.3.3 Analytic versus synthetic approaches

This example brings out some interesting points:

- In a given entity, identifying denoted and meta levels is merely a question of viewpoint, which depends heavily of the context. Each context defines a precise viewpoint whose reification makes local some information.
- When changing 1 entity for n, reflection consists usually in making explicit some hidden aspects of the denotation's environment. This operation is analytic by nature.
- When changing n entities for 1, reflection consists usually in hiding some irrelevant aspects of the denotation's environment. This operation is synthetic by nature.

The main stream in reflection favours individual approaches: any object at a level has its own unique meta-object at the next level [24] [30]. Nevertheless, organizational reflection provides a framework suitable for both analysis and synthesis, supporting different viewpoints, as well as flexibility and modularity needed for the implementation of adaptive computer systems:

- the analytic approach reifies the components and their relations, opening the door to internal restructuring (suppression, addition or replacement of any component or relations) like surgery or message stream re-routing would do.
- the synthetic approach decreases the number of entities and, therefore facilitates the interaction management: it is usually easier to deal with the interactions among 4 entities than it is for 4000.

3.3.4 Why multiple viewpoints ?

Can there be several meta-actors attached to the same actor ? It is easier to handle both reflective computations and infinite tower problems[1] when there is a single meta-actor. However, such limitation might tend to shift narrowness and brittleness from the actor to the meta-actor: a sole viewpoint would be embodied in the meta-actors from bottom to top of the tower. In return, several meta-actors would allow different or even divergent viewpoints to coexist. Such perspective changes are as well discussed by Winograd and Flores [32] who think they are necessary for adaptation.

ReActalk, in its present state, proposes two meta-actors per actor: the first one analyzes the processing of external stimuli (messages) by the individual actor's behavior, the second one reifies the organization (ecosystem) in which the actor is embedded[2].

3.4 On the permanence of changes

Since adaptation leads to self-modifications, a natural question regards the permanence and the range of changes. Which factors should decide that a change is definitive or that an actor should go back to its initial state ? Should other actors belonging to a same class be touched by one's modifications ? Again [Bateson, 1979] provides insights, distinguishing between genetic (changes applied to a whole species) and somatic control (changes applied to a single individual). Genetic control is more economic in terms of somatic adaptation mechanisms, but is also more rigid and lacks reversibility. On the other hand, somatic control is flexible and reversible, but expensive in terms of adaptation mechanisms.

ReActalk translates somatic changes by changes within metalevel hierarchy, and genetic changes by changes within class hierarchy. This leads to a reformulation of the permanence problem:

> when should a modification at the meta-actor level be integrated in the class hierarchy, and therefore be applied to a class of actors instead of just one ?

Bateson gives a partial answer:

> *At first glance, there are those cases in which the flexibility would perhaps never be needed after the shift to the genetic. These are cases in which the somatic change is an adjustment to some constant environmental circumstance Those members of a species that are settled in high mountains may as well base all their adjustments to mountain climate, atmospheric pressure,and the like on genetic determination. They do not need that reversibility which is the hallmark of somatic change. Conversely, adaptation to variable and reversible circumstances is much better accomplished by somatic change, and it may well be that only very superficial somatic change is tolerable.* [3, p. 170-172]

Through meta-actors and class hierarchies, ReActalk supplies the tools to explore both genetic and somatic control.

3.5 On the nature of metalevels: infinite versus indefinite tower

> [...] *it seems probable that in most instances of learning and somatic changem, the number of levels of somatic control is small. We can learn and learn to learn and possibly learn to learn to learn. But that is probably the end of the sequence.* [3, p.178-179]

Since there is nothing in reflection that decides what is to be reified and what is to be absorbed [27], we claim that reflection is suited for adaptation. Furthermore adaptation is not based on *infinite* towers of reflective metalevels, but rather on *indefinite* ones. Lazy creation of metalevels in ReActalk gives the opportunity to adapt the viewpoint according to the immediate circumstances. In spite of the elegance and the attraction of infinite towers, they support a sole viewpoint, is transmitted from one metalevel to the next one. Since adaptation depends heavily on the context, this is a handicap. The fluidity of the real world makes it hard to know *a priori* what will be the vantage point for adaptation.

[16] discusses the pros and cons of three different types of meta-meta-actors, according to different standpoints on the meta-actor and its role. The three perspectives proposed are adaptability, concurrency and information flow. Recall that the meta-actor which takes care of the actor's self-representation, is described as an organization in charge of the actor's message-processing. Further steps of the construction of the tower raise the question of the meta-actor's self-representation. Traditional solutions adopt uniformity as a principle, with the conclusion that both actor and meta-actor share the same nature. Applied to the meta-meta-actor, the same principle implies that this one is an organization in charge of the meta-actor's message-processing.

However, different perspectives may be convenient depending on the context:

---

1 To be reflective, a system requires, at least, a representation of itself, fit for manipulation and modification. This representation is known as *metasystem*. Moreover, any modification of the metasystem have an immediate effect on the behavior of the system itself; this link is known as *causal connection* between a system and its metasystem. Although it is tempting to apply to the metasystem the same treatment, and thus to give it a metametasystem, this approach leads to an infinite regression known as the *infinite tower*: the metasystem is described by a meta-metasystem which itself is described by...

2 There is a third one inherited from Smalltalk-80: the actor's class reifying its structure.

- with the accent on adaptability, the meta-meta-actor views the meta-actor as an organization, as illustrated on Figure 1;
- with the accent on concurrency, the meta-meta-actor views the meta-actor as an actor, leading to architectures such as ABCL/R2 [25].
- with the accent on information flow, the meta-meta-actor views the meta-actor as an "enriched" Petri net.

3.6 Somatic adaptive processes implemented within ReActalk.

In this section, we present some adaptive mechanisms implemented in ReActalk [14]. They help an agent to adapt to other agents by modifying its model of computation. So they are devoted to somatic adaptation.

An agent cannot elude stimuli arising from its ecosystem. Within ReActalk, stimuli are messages. An agent's ecosystem consists of the agents it has to cooperate with. Adaptation makes cooperation easier, smoothing interactions between heterogeneous agents.

A newborn agent operates according to one of the predefined messages interpreter. Such an interpreter embodies part of messages semantic. For instance, in the implementation of ABCL [33], the prefix now calls for a synchronous communication, whereas the ACTORS model of computation [1] does not support this protocol. Thus by means of messages, an agent may ask for specialized protocols for cooperation, possibly unknown to the addressee. When an agent does not understand a message, a reasonable interpretation is to ascribe the failure to its interpreter. Modifying the interpreter could be the appropriate solution. In this scheme, breakdown-driven adaptation leads to the modification of the agent's self.

Through reification, reflection explicits an agent's interpreter. Within ReActalk, abilities inherent to an interpreter are abstracted in genes. A gene describes a partial organization of agents devoted to a specific aspect, e.g. the agents and relations necessary to implement the semantic of the prefix now. The complete organization acting as the interpreter is assembled from such genes. In order to achieve this assembly, we developed a mechanism called hybridization. Hybridization, instead of simply composing organizations, literally merges them together: solely the most specialized agents and relations are kept. The hybrid organization usually exhibits a combination of the abilities of the parent organizations, but sometimes new abilities emerge.

Therefore, when a message is not understood, an agent consciously triggers adaptive mechanisms. This triggering is implemented with the exceptions handling system of Smalltalk-80. Firstly a signal is raised and the interpreter is reified. The faulty message is then used to search for genes able to interpret it. Hybridization dynamically proceeds and modifies the agent's interpreter. Finally the faulty message is sent again. In that case, adaptation mainly relies on analysis, i.e. the decomposition of the self into an organization.

Conversely, in order to compensate the fattening effects of hybridization, we have also implemented the inverse operation, namely deshybridization. It allows an agent to get rid of obsolete somatic changes. Deshybridization is unconsciously activated by a surveillance process grafted at the metalevel. Deshybridization is completed by a reflect operation which annihilates the reification.

Thus an agent adapts its somatic abilities according to the agents with which it interacts and these abilities are removed when they are no more necessary. In this way, somatic adaptation smooths heterogeneity.

4 Adaptation through modification of one's environment

Section 2 mentioned adaptation through modification of one's environment as an alternative to adaptation through change of one's behavior. ReActalk supplies as well the tools for the exploration of this avenue.

On the one hand, an actor can act on its environment by communicating with the actors located at its own level. The environment is perceived as an ecosystem, whose components are members and relations. Organizational reflection makes the reified ecosystem accessible to any of its entity. Thus, an actor can know about the others in the same ecosystem.

On the other hand, an entity can act directly upon its environment structure, now accessible. Organizational reflection applied to ecosystems combines reification and causal link to actualize the modifications made to the environment by one of its entities.

5 Related works

Several research projects apply reflection to actors and/or organizations. Probably the first to explore reflection in concurrent programming is ABCL/R [29]. ABCL/R exemplifies a trend where every actor has its own meta-actor which centralizes reified aspects (both structure and computation). ReActalk breaks away with this trend in allowing multiple metarepresentations for an actor:

- its meta-actor : a metarepresentation expressing the actor's autonomy by reifying computations;
- its class : a metarepresentation that regards the actor as a structure;
- its reified ecosystem : a metarepresentation expressing the agenthood of the actor.

So these metarepresentations reify in a causally-connected way a standpoint on the actor.

Furthermore unlike ABCL/R where the reification of computations lies in the the meta-actor's script, within ReActalk the reification of computations is distributed

through an organization taylorizing message processing. Therefore when reflective computations proceeds from within the meta-actor through the script in ABCL/R, reflective computations are performed through organizational modifications and the management of messages is done by agents in the ReActalk.

Mering IV [11] is a reflective system for DAI that also supports multiple metarepresentations for an actor: one in charge of the structure reification and one in charge of the computations reification. Reflective message management relies on hard-wired reflective methods. Nonetheless these infinite towers focuses on the actor taken in isolation, occulting organizational questions. As in ABCL/R, it is not possible to change the number of entities taken into account for reflective shifts from level to metalevels, as ReActalk can do.

Due to the lack of a global view on computations (coordination of groups of actors and resources sharing), ABCL/R2 introduces the group concept within ABCL/R and applies reflection on such groups. The brand-new meta*-groups are essentially concerned with computation (evaluation, scheduling...). A metagroup is made of

- the meta-objects of the base level objets of the group;
- the group kernel objects (group manager, evaluator and meta-object generator) which are the causally-connected self representation for the structure and computation of the denoted group;
- other metalevel objects, e.g. a scheduler

In ABCL/R2, there is no reification of interactions as ReActalk does through communication protocols. We believe that communication protocols are important because they reify interactions, a key concept for organizations and ecosystems, thus forming a privileged lever to actualize and manipulate the rules regulating an ecosystem. In fact, meta*-groups in ABCL/R2 are exclusively concerned with the management of computational resources, e.g. processors, by coordinating among the meta-objects of the members and the group kernel objects. In ReActalk, the reification of ecosystems is a keystone to adaptive behaviors. For instance, under ReActalk philosophy, a processor is simply a base level agent within a "computational" ecosystem and its use is regulated by the rules made explicit within the reified ecosystem. Therefore the way of modelling systems is very different from ABCL/R2 and the reification of ecosystems goes far beyond the notions of groups and metagroups embedded within ABCL/R2.

6 Open problems and future works

Respect to the fitness of ReActalk towards adaptation, many paths still remain open. This section briefly exposes some: multiple self-representations, genetic versus somatic control and environment modification.

6.1 On the cohabitation of multiple perspectives

As pointed out in [26], there are many notions of self. Furthermore reified aspects and absorbed aspects depend on the context and the intention. Many systems are designed to support multiple perspectives on the self [20] [9]. ReActalk proposes and implements two viewpoints: individual (structure and computation) and organization.

To go further, ReActalk should be provided with a mechanism bound to integrate multiple perspectives. The combination of self-representation and multiple perspectives on one's self lead to a coreferential approach [5] of metalevels for tightly coupled systems. The micro-theory concept [17] would be helpful for weakly coupled systems where some incoherences may be tolerate.

6.2 Genetic versus somatic control

Through class and meta-actors hierarchies, ReActalk supplies the tools to explore both genetic and somatic control. It is then a matter of seeing how the related reflective towers can model such kind of control.

6.3 Adaptation through modification of one's environment

In order to adapt, an individual can change its environment's reification. Due of the causal link, it modifies as well the environment itself. This leaves several questions unanswered:

- what is the nature of admissible changes ?
- who is allowed to make these changes ?
- when and how to proceed in order to maintain some coherence ?

Maybe the answers will teach us a lot on the modification of our own environment.

6 Conclusion

This paper has presented the design principles behind ReActalk: ReActalk uses an organizational approach of reflection and is meant to provide a framework for the study of adaptation in the context of agents and OS. Since OS interact with a fluid reality, they must adapt in order to mitigate the consequences of the narrowness and the brittleness of traditional systems. Accordingly, departing from the trend that uses reflection to provide users with hooks to open and customize the programming language and environment, ReActalk uses reflection as a framework for adaptive processes evolving in a dynamic environment.

We have shown, on the one hand, how reflection can implement adaptive processes, and, on the other hand, how adaptive mechanisms can be actualized in ReActalk. We have pointed out two ways of performing adaptation,

based on Gregory Bateson's work: adaptation, viewed as a modification of one's behavior to circumstances, and adaptation, viewed as a modification of one's environment. We have also outlined how to get from Bateson's view on adaptation to an architecture suited for adaptation. Finally, we have shown that the organizational approach of reflection implemented in ReActalk allows both types of adaptation.

7 Acknowledgments

We are indebted to Mr Brian Smith and Mr Les Gasser for their insightful comments. Our discussions with them have been the source of many changes, refinements and even more, deeper understanding.

8 References

[1] Agha, G., *Actors: A Model of Concurrent Computation in Distributed Systems*, MIT Press, 1986, 144 p.

[2] Bateson, G., *Vers une écologie de l'esprit*, vol. 1-2, Paris, Éditions du Seuil, 1977.

[3] Bateson, G., *Mind and Nature: a necessary unity*, Toronto, Bantam Books, 1979.

[4] Beer, R. D., *Intelligence as Adaptative Behavior*, New York, Academic Press, 1990, 213 p.

[5] Bourgeois, R., *ICEO : Intension, Coréférence et Objets dans la fédération de formalismes de spécification*, Université Paris 6, France, 1990.

[6] Bouron, T., J. Ferber and F. Samuel, *MAGES: a Multi-Agent Testbed for Heterogeneous Agents*, in *Decentralized A.I. 2*, Demazeau, Y. and J.-P. Müller eds, Amsterdam, North Holland, 1991, pp. 195-214.

[7] Briot, J.-P., *From Objects to Actors: Study of a Limited Symbiosis in Smalltalk-80*, LITP 88-58 RXF, Université Paris 6, 1988, 52 p.

[8] Brown, P. J. *Throw-away Compiling*, Software - Practice and Experience, vol. 6, 1976. pp. 423-434.

[9] Ferber, J., *Objets et agents: une étude des structures de représentation et de communications en Intelligence Artificielle*, thèse de doctorat d'état, Université Pierre et Marie Curie, June 8, 1989, 492 p.

[10] Ferber, J., *Eco-Problem-Solving: How to solve problems by interactions*, report 05/90, LAFORIA, Institut Blaise Pascal, France, January 1990, 18 p.

[11] Ferber, J. and P. Carle, *Actors and Agents as Reflective Concurrent Objects: a Mering IV Perspective*, IEEE Transactions on Systems, Man, and Cybernetics, 1991, vol. 21, no 6, pp. 1420-1436.

[12] Gasser, L., *Social Conceptions of Knowledge and Action: DAI Foundations and Open Systems Semantics*. Artificial Intelligence, 1991, vol. 47, no 1-3, pp. 107-138.

[13] Gasser, L. and T. Ishida, *A Dynamic Organizational Architecture for Adaptive Problem Solving*, AAAI-91 Proceedings, July 14-19 1991, pp. 185-189.

[14] Giroux, S., *Agents et systèmes, une nécessaire unité*, Ph. D. thesis, University of Montreal, 1993.

[15] Giroux, S. and A. Senteni, *Reactalk, a Reflective version of Actalk*, in [19], 5 p.

[16] Giroux, S. and A. Senteni, *Taylorizing the Behavior of an Actor: an Organizational Approach of Computational Reflection*, East EurOOPe Proceedings, September 17-20, Bratislava 1991, Czecho-Slovakia, SIGS Publications, Inc. New York, pp. 12-26.

[17] Hewitt, C., *Open Information Systems Semantics for Distributed Artificial Intelligence*, Artificial Intelligence, 1991, vol. 47, no 1-3, pp. 79-106.

[18] Ibrahim, M. H., *Informal Proceedings of ECOOP/OOPSLA '91 Workshop on Reflection and Meta-level Architectures in Object-Oriented Programming*, Ottawa, Canada, Oct. 22, 1990.

[19] Ibrahim, M. H., *Proceedings of the OOPSLA '91 Workshop on Reflection and Meta-level Architectures in Object-Oriented Programming*, Phoenix, Arizona, Oct. 7, 1991, Xerox Technical Report, 1991.

[20] Ishikawa, Y., and H. Okamura, *A New Reflective Architecture: AL-1 Approach*, in [19] 5 p.

[21] Kiczales, G., J. des Rivières and D. G. Bobrow, *The Art of the Metaobject Protocol*, The MIT Press, 1991.

[22] Langton, C. G. ed., *Artificial Life*, Redwood City , CA, Addison Wesley, 1989, 655 p.

[23] Launchbury, J., *A Strongly-Typed Self-Applicable Partial Evaluator*, 5th Conference on Functional Programming Languages and Computer Architecture, Aug. 1991, Springer-Verlag, LNCS-523, pp. 145-164.

[24] Maes, P., *Concepts and Experiments in Computational Reflection*, OOPSLA '87 Proceedings, Orlando, Florida, October 4-8, pp. 147-155.

[25] Matsuoka, S., T. Watanabe and A. Yonezawa, *Hybrid Group Reflective Architecture for Object-Oriented Concurrent Reflective Programming*, ECOOP '91, July 15-19, Geneva, Switzerland, Springer-Verlag.

[26] Smith, B. C., *Varieties of Self-Reference*, 1986 Conference on Theoretical Aspects of Reasoning about Knowledge, Monterey, CA, pp. 19-43.

[27] Smith, Brian C., *What do you mean, meta?* in [18].

[28] Wand, Y. and Carson W. C., *An Approach to Formalizing Organizational Open Systems Concepts*, Conference on Organizational Computing Systems, Nov. 5-8, 1991, Atlanta, Georgia, SIGOIS Bulletin, vol. 12, no 2-3, pp. 141-146.

[29] Watanabe, T. and A. Yonezawa, *Reflection in an Object-Oriented Concurrent Language*, OOPSLA '88, San Diego, CA, Sept. 25-30, vol. 23, no 11, Nov. 1988, pp. 306-315.

[30] Watanabe, T. and Akinori Y., *An Actor-Based Meta-level Architecture for Group-Wide Reflection* in [18].

[31] Watzlawick, P., J. H. Beavin and D. D. Jackson, *Une logique de la communication*, Éd. du Seuil, 1972.

[32] Winograd, T. and F. Flores, *Understanding Computers and Cognition: A New Foundation for Design*, Addison-Wesley, 1987.

[33] Yonezawa, A. eds., *ABCL An Object-Oriented Concurrent System*, The MIT Press, 1990, 329 p.