Human Cooperation Improvement Using Autonomous Skill Agents

Olivier Chator
Conseil Général de la Gironde
IMS Laboratory, CNRS, IPB
Université de Bordeaux, France
o.chator@cg33.fr

Pierre-Alexandre Favier and Jean-Marc Salotti
Ecole Nationale Supérieure de Cognitique
IMS Laboratory, CNRS, IPB
Université de Bordeaux, France
pierre-alexandre.favier@ensc.fr, jean-marc.salotti@ensc.fr

***Abstract*— A local authority, the "Conseil Général de la Gironde" in France, manages various projects in different fields, like sustainable development, and coordinates public and private partners' actions. The observation shows that each of them has only a partial vision of others' skills and know-hows. Generally speaking, everywhere where human collaboration is needed, sharing skills is one of the problems identified. This work addresses these difficulties using a learning and collaborative multi-agent system to enhance skill sharing and management. One of the main innovations here is that skills are represented as autonomous agents, and not just as capabilities, as is usually the case.**

***Keywords*-Multi-agent systems; sustainable development; skills; governance**

I. INTRODUCTION

A local authority, the "Conseil Général de la Gironde" (CG33) is responsible for public actions for 1.5 million inhabitants. Numerous domains are concerned: school transportation, management of middle schools, tourism development, solidarity, integration and support for elderly people. One of the CG33 missions is to define policies and practices for the Sustainable Development (SD) of the department (a territorial division lower than regions). For example, the objective could be to transform a neighborhood into an eco-district [1, 2]. Experience shows that this type of project is very complex and requires the collaboration of many public and private actors under the supervision and management of a "project supervisor" (PS), for instance an architectural firm. Each actor has only a partial knowledge of the capabilities of the other and some information is sometimes lacking, but the PS has to take decisions anyway. In addition, the objective is often to minimize the costs and to obtain energy or ecological labels, which typically are antagonist objectives. For the PS, it is often difficult to understand the impact of each parameter. The preferred option is usually the one that is better understood, which comes at the expense of other options because there was insufficient knowledge on their impact, cost, and implementation. In order to help the actors, and especially the PS, to find the best partners, the CG33 decided to build a database of skills and actors [1]. For example, it should be possible for a PS who wants to renovate some buildings to identify skills and actors in various domains such as thermal insulation, thermal simulation, air tightness, and installation of different types of photovoltaic panels on the roof. In turn, the partner who has expertise in thermal insulation may require the help of another partner who is specialized in the use of specific insulation materials. Thus, the challenge is here to allow each stakeholder of an SD project to share and learn more about the expertise and know-how of the others. In general, whatever is the field of activities (e.g., building the best sport team as possible), everywhere where human collaboration is needed, the problem is the same. Therefore, to facilitate the increase of skills for each actor over time and to stimulate their cooperation, building an efficient system of skill sharing is the key.

A traditional approach could be to build a simple database with a direct link between actors and skills. However, considering the central role of skills and the needs for constant evolution and modifications of the data, a research project has been carried out in our laboratory to find and implement a better solution. It is suggested here that a multi-agent system (MAS) is more appropriate. In our proposal, the innovative key concept is to consider each skill as a "full agent", and not only as an agent's ability as is usually the case [3, 4, 5]. Having their own learning model and their own life cycle, our skill agents are autonomous, cognitive, and they interact with human actors to stimulate and improve cooperation. Finally, this work offers a proposal to improve the management of skills, to make them more efficient, during projects, and over time. If a user wants to address a new goal, how should he define the project and what are the skills needed to implement it? How can he get benefits from past projects?

The model is described in Section II. Some results are shown in Section III and Section IV concludes this document.

II. MODEL

A. The Issue

Let us introduce the issue with an example of an SD project that aims to "transform a neighborhood into an eco-district". Let us assume that a PS has to build a HQE building. HQE stands for "Haute Qualité Environnementale" and is a standard for green building in France [6]. For this kind of project, the PS needs:

- A definition of the goal to achieve (the objective)
- Skills such as "integrating insulation materials" in order to meet the HQE objectives.

- Actors such as private building companies to implement the skills.

Using our "SD skill sharing" system, the PS should be able to identify a list of possible partners. Intuitively, we might think that this list could be simply sorted according to the most experienced partners for the given task. However, other criteria than just experience should be taken into account: price, quality, duration, localization, expertise with specific materials, etc. The system may suggest a partner according to this list of criteria. In addition, it also has to select different companies over time. The problem is to determine a good strategy in order to achieve that goal throughout a project.

Going further, it is also interesting to get the benefits of past experiences on similar projects over time, to build new ones. Let us illustrate with the objective: "*I want to put photovoltaic panels on the roof of my house*". I have to find the skills required for this new project. Taking my needs and experience of past projects into account, there are two possibilities:

- I find a past project that reflects exactly what I want to do. Thus, what I need is to find a way to retrieve all the skills of this project, and proceed to my new project creation using this list.
- I find a past project, but it is not exactly what I want to do. Thus, what I need is to find a way to retrieve some of the skills of this project, and proceed to my new project creation using this restricted list.

The two points above are efficient if the user finds projects that already contain all or some of the skills he needs to build the new one. However, this is not always the case. Skills may be scattered throughout various projects. Thus, the point is to find a way to answer a limited expression of needs at specification level. For example, if we consider that in the system we already have the past two projects: "*wind turbine implementation*" and "*hydraulic micro power implementation*". They both belong to the same "domain": "*new means of energy production*". If the user wants to build a new project, in the same domain (e.g., the implementation of solar panels), an interesting idea is to look for skills used in all the past projects of this domain. Thus, the system will suggest integrating the skills that were fluently used across all projects in the domain.

It is possible to generalize from this example. Each professional sector of activity has procedures and processes, each of them used to reach specific goals or objectives. Each objective may be implemented through some projects and skills. Generally speaking, it may be difficult to make processes evolve according to environmental constraints. When successful, the knowledge and know-how that have been used should be capitalized on for possible use in further projects. Finally, the main problem is to find a way to improve the management of skills to make projects more efficient over time. To do this, it is possible to build new projects, working on past projects or objective domains. Thus, the goal of this work is to improve new project definitions dynamically and to identify all the skills needed to make them a success.

B. Defining a Skill

A skill is the ability to exploit some knowledge and know-how in order to solve a class of problems. It is different from a competency, which is generally accepted as a set of behaviors or actions needed to be performed successfully within a particular context [4]. In this study, for the sake of simplicity, it is assumed that a skill is a sum of elementary competencies (ECs).

The main specifications of our application are to store information about the skills of possible participants in SD projects and to suggest interesting partners for a given skill. An important issue is to make the link between observations (e.g., "partner A has been assigned the role of task 1 and 2 in project X and has succeeded in implementing solutions") and skills, which do not correspond to the names of the task. Let us present an example:

Integrating glass wool for the insulation of northern walls in a specific building in a given project is related to the skill "integrating insulation materials". However, integrating isolated wood panels under roofs might be very different from integrating glass wool in walls and the best expert for the first task might not be the best one for the second. The skills might be differentiated by small details, but, for the proposed application, it would be irrelevant. It is expected that the users of the application will ask general questions such as "who has skills in insulation materials?". The key problem is to find the appropriate level of detail for each skill and to make the difference between an elementary competency that belongs to a skill and the skill itself. Then, assuming that a skill is defined at the right level and includes a list of possible elementary competencies, the question is to determine how each of them participates in the definition of the skill. For instance, for the skill "integrating insulation materials", how important it is to have the know-how for isolated wood panels? In other words, there should be an associated weight for each elementary competency and there should be a mechanism for learning them integrated in the skill agent

According to the needs of the project, a skill can be created at any moment, its definition (the list of elementary competencies) may evolve, it can eventually be split into different skills and it might even be removed. Such constraints cannot easily be handled in a standard database in which the actors and their skills would be stored. Because of the central role of the skills, it is suggested in this paper that the skills be considered as agents of a multi-agent system. However, in most applications, agents are associated with models of actors in the real world, the skills defining the behavioral rules [2, 7, 8]. The problem is that the skills have their own dynamics and are rather independent from the actors. The skills should be agents with their own lives. In addition, if the skills and the actors are distinguished, it is difficult to define actors as other agents of the system. In cognitive science, the embodiment of mind is often considered a requirement to obtain an effective agent [9, 10, 11, 12]. Skills alone have no perception, no motivation and no means to perform an action and change their environment. Nevertheless, it is possible to define these

elements artificially. Intuitively, a skill can be motivated by the improvement of its own definition, e.g., a weighted list of elementary competencies and the clarification of its relationships with the other skills. The user of the system has another motivation: he wants to find a partner for his project. The system should provide some criteria and suggest an actor for the required skill. The user makes his choice, then the work is carried out (embodiment of the skill) and an evaluation of the realization is performed. The key idea is to consider that a criterion is no more than an abstraction of a hidden list of elementary competencies. For example, the duration of a work is not a competency. However, implicitly, it is closely related to the ability to work fast, which is an elementary competency of the skill. Therefore, the skill can exploit the definition of the criteria, which evolve according to the evolution of the projects, to characterize its definition. Another issue concerns the links between the skills. Different skills may have several elementary competencies in common. If no actor is found for a given skill, an interesting idea is to make suggestions with actors associated with the skills that are closely related.

In addition, a database is still required for the storage of past observations (e.g., Actor A has been involved in project X for the embodiment of skill S with an evaluation of a list of criteria C_1...C_n).

C. Definitions and key concepts

1) Main concepts

We can use another concrete example to illustrate those concepts: "*To make energy savings, I want to put better insulation into the walls of my flat*". From this example, we may define four concepts in our proposal.

a) Environment

An environment is viewed as a professional sector of activity. In our example, "*sustainable building sector*" is the environment in which the user request occurs. This is the highest level of abstraction and it is related to the professional sector of activity.

b) Objective domain

An objective domain is a group of objectives, concerned by the same theme of activity. In our example, "*thermal insulation improvement*" is the objective domain in which the user request occurs. Another domain could be "*air tightness improvement*". The idea is to position objectives within one or several objective domains.

c) Objective

An objective is a simple textual description of a goal to be reached (NB: the term "goal" would have been more appropriate, but "objective" was chosen from the start for convenient reasons and links with the French language). In our example, the objective is to "*put better insulation into the walls*". This objective is part of the "*thermal insulation improvement domain*". Considering this simple question from the user, no constraint about the materials or skills used to reach the goal is expressed. Another objective could be "*improve air tightness in my flat*". In our implementation, an *Objective* agent is available. To transpose an objective into "real life", it is mandatory to define a project first.

d) Projects

A project is defined by an objective, a start date, an end date, resources (like human actors) and processes to schedule the list of skills to be used. In our example, a project defined by the "*integration of glass wool into walls*" is proposed. It will start next week, will stop in 15 days, and requires several skills. Finally, the project is implemented and evaluated at the end using ECs of each skill.

2) Types of Objectives

For a user, the problem is to reach a goal. It is usually defined by a simple assertion like "*I want to do something*". In order to reach the goal, the user will define a new project in our system. He does not often have knowledge of all the skills that have to be used in the project. As it is difficult to give a unique answer, depending on the user request at time T, two approaches are proposed for building new projects. The first one is based on completed projects and their objectives. The second consists of building the new project using only concerns about the objective's domain, and possibly the environment. This case occurs when the user wants to do actions in a particular domain of activity, but does not know exactly what to do.

D. Skill Agents: theoretical proposal

1) Introduction

Generally, an agent has behaviors. Each of them could be presented as shown in Figure 1.

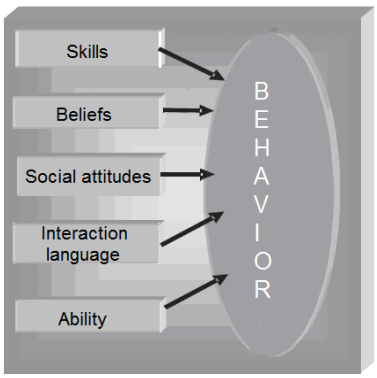

Figure 1. Agent's behavioural characteristics

It is assumed here that a skill is unique and can be implemented as an agent in a multi-agent system. It has resources (a list of physical actors) and its own life cycle. It can be created, can evolve and can eventually be removed when not used anymore or when replaced by another skill agent. Skills agents fit into a multi-agent system, where the environment is defined by the interactions with the users. They are cognitive, non-conversational and non-dialogic [3, 5]. They never directly communicate with human users. They react and evolve according to information

modifications and requests from the user via a *WebRequester* agent. An important feature is their ability to learn how to define themselves and how they are linked to the other skills.

2) Definition

Skill agents are defined by three main features: perception, internal attributes and actions:

- **Perception**: Skill agents are listening to information broadcast by the system after interaction with the users. It can be, for instance, an update after external observations (e.g., a new project is started or the result of work for a given project is inserted in the database) or a request is sent by another agent within the MAS.
- **Internal attributes**: A skill agent is determined by the list of elementary competencies that defines the skill, a creation date (appearance in the MAS), a domain(s) of activity and a specific "*age*" (see below). It also has a list of behavioral rules, expressed in XML format with a specific grammar (see below).
- **Actions**: If there is an update of an external observation that is linked to the skill, the agent updates its database and its weights according to a learning rule. It provides an answer to the *WebRequester* agent (which, in turn, informs the user) according to a strategy defined by behavioral rules. Each skill agent has the ability to establish links with other skill agents in the MAS. This last action is based on its environment analysis, automatic (or not, if specifically requested by the user), and defined in its behavioral pattern. More importantly, skill agents are proactive. When a user wants to create a new project (typically a new objective action), an *Objective Agent* receives those requests. Then, it sends a broadcast to inform all skill agents. Each of them determines if it is a candidate for participation (or not) in the new project. The decision is defined through the computation of what we call a "proximity coefficient" (see below). At the end of treatments, the *Objective Agent* returns a list of candidates to the user for (in)validation according to the project needs.

3) Life cycle

Skill agents have their own life cycle, divided into 3 "ages", according to their specific levels of autonomy.

1. **Childhood**: The skill agent runs in a "learning" mode. During this age, the aim is to make the agent "grow". When it is created, the first step is to assign to it a list of criteria (elementary competencies) for future evaluations. The second step is to associate a list of actors. At initialization time, there is no evaluation in the database because the agent has not been used yet. Thus, if a user looks for an actor (like a rugby player) for this skill, the agent is not able to make relevant suggestions (childhood). It simply returns a list of potential actors ranked according to the number of times they have been involved in realizations (the most experienced at the end). Once the result of the work is available, the user evaluates the criteria associated with the skill and the data are stored in the database. At this age, the skill agent does not really communicate with other agents and uses only basic behavioral rules. It grows until it reaches a threshold of evaluations (e.g., after 3 concretizations across various projects) in the database. When this threshold is reached, its age automatically grows to the next one.
2. **Teenage**: When a user looks for an actor with this skill, the agent computes a list of candidates, exploiting the previous results (past experiences) and the current user criteria weights values. If the user does not specify any weight for criteria, default values are used (e.g., weight=1). A list of potential actors is then obtained after a dynamic computation of criteria weights (see subsection D.2). At this age, the autonomy of the agent is rather limited. It communicates and tries to build relationships with other agents (see subsection D.3). Whatever the action performed, a human validation is requested and stored in the skill agent memories table in the database. The skill agent grows until it has a threshold of evaluations (e.g., after 10 concretizations across various projects) in the database. When this threshold is reached, its age automatically grows to the next one.
3. **Mature:** The skill agent is able to make direct and relevant suggestions to the user as soon as a project is created. A list of skills is proposed with the possible actors for each of them. Obviously, the user can still make modifications but he can save a lot of time if the choices correspond to his needs. At this age, the agent has a good knowledge of its relationships with the other agents. Using the system parameters, it is possible to cancel the validation of the system choices by humans. For example, in our SD skill sharing system, we could imagine characterizing a new project (e.g., building a new middle school) without knowing what the skills needed are, nor the actors who are able to implement them. In this situation, at the mature age, the system will automatically choose the skills (using the links between them) and will affect their concretization to actors. Thus, the skill agent is

fully autonomous, makes its own decisions, and does not need human validation.

Remark: the thresholds for age transitions have been set empirically. The objective is to grow as rapidly as possible. It is a trade-off between giving a help to the user as soon as possible, and reaching a high level of expertise.

E. Learning mechanisms

1) Introduction

In the literature, we can find various types of learning mechanisms for agents: the Markov Decision Process [13] with reinforcement learning [14], the theory of games (matrix games [15] and stochastic games [16]), the Bayesian networks [17], the Case-Based Learning (CBL) methods [18], and so on. Our skill agents evolve according to human actions on the system (requests, selections, validations, and evaluations). In our context, the CBL methods seems more adapted because they are based on valuated and memorized iterations of concrete experiments, they integrate validations of human on the system decisions, and allow (for a given environment) specific and gradual adjustments over time. However, existing CBL algorithms [19, 20, 21, 22, 23, 24, 25] are not entirely appropriate to our problem because the objective is not to find a similar case in the knowledge database. Selected actors will have the skill anyway. It is rather to make a choice among several possible actors according to a global skill sharing policy.

Therefore, synthetically, learning mechanisms for a given skill agent, are based on two main points: actors' selection mechanism and the building of links with other skill agents.

2) Actor selection

When the user needs to find an actor for a given skill (generally inside a project), he asks the system for suggestions. We have already seen that a skill is defined by a weighted list of criteria or elementary competencies.

The first step is to give this list with undetermined (or default) weights to the user. For instance, in an SD activity domain, if the user wants to retrieve an actor for the "thermal insulation" skill, the system asks the user to define the weight associated with each criterion: "wall insulation", "roof insulation", "wood based materials", "diagnostic", "price", "duration", etc. This information is used to update the definition of the skill.

Assuming the user gives specific values to the weights of each criterion, the second step consists in computing the new weights. This is done using those new values plus the old validated computed weights values. "*Validated*" means approved by the human user in past experiments. Let **k** be a criterion and $\mathbf{W_{k(t)}}$ the weight associated to it for request number **t.** For **t=1**, the average weight is set to $\mathbf{W_{k(1)}}$ which is the weight given by the user. For **t > 1** the new average weight is computed using equation (1).

$$\overline{W}_{k(t+1)} = \frac{(t \times \overline{W}_{k(t)}) + W_{k(t+1)}}{t+1} \quad (1)$$

The third step is to build the list of actors, evaluating them. The proposal is to use the new computed weights of each criterion, and the evaluation results of previous realizations (experiments) of the skill. Let **f** be the number of times where an actor **a** has concretized the skill in past projects. Firstly, we compute a partial value using the old "after work" evaluations (equation (2)):

$$\overline{E}_{k(a,t)} = \frac{\sum_{j=1}^{f} E_{k(a,j)}}{f} \quad (2)$$

Secondly, the "actor evaluation" is done with equation (3):

$$Eval(a) = \sum_{k=1}^{t} \overline{W}_{k(t)} \times \overline{E}_{k(a,t)} \quad (3)$$

Please note: if the user has given specific values for the criteria weights, we will have $\overline{W}_{k(t+1)}$ instead of $\overline{W}_{k(t)}$.

In order to give a chance to each actor, we propose to alternate between the performance policy (linked to evaluations) and the skill sharing policy (a random selection process), with a probability of 0.5. Thus, an actor with systematic lower evaluations is not penalized. Additionally, in any case, the user can still select an actor who is not at the top of the list. Considering the skill agent life cycle (as seen in Section III), a human validation of the system choice is needed when the skill agent is in childhood or teenage age. When it is mature, the validation is considered as implicit.

The validation process consists in updating the skill agent memories, typically a record in a table, where the current computed weight $\overline{W}_{k(t+1)}$ of each criterion was stored. When the choice is approved, by the user or automatically at mature age, those weights become validated $\overline{W}_{k(t)}$. Being now part of what we called "*old validated computed weights values"*, they will be used for the next $\overline{W}_{k(t+1)}$ computation. Such a method ensures a learning activity, reflects "real life" situations, and is really close to what is done in Case-Based learning systems.

3) Agent links

For some projects, a user who is not experienced may not necessarily know all the skills that are required for the realization of a project. Concrete example: "I am working in a local authority, in a little town, and would like to organize a public event to stimulate sustainable behaviors around waste sorting in my citizens". In "real life", this is a case of use of our SD skill sharing system in Gironde. To solve my problem, I can ask the system to help me and suggest a list

of skills in order to succeed. The problem is the same if I am the coach of a rugby team: "I would like to organize some specific training for my players at such a position. What do I have to do, who can help me?"

In a MAS or agent's point of view, in order to provide an efficient answer, it is possible to exploit the links that can be found among the skill agents that are at least in the teenage age. For a given skill agent S, the proposal is to build links with other skill agents by computing for each of them a "proximity coefficient" (in percentage) to S. This coefficient is built using what we call "similarities" to S that are sought in the MAS environment. Similarities are found by searching for elements that S and the other skills have in common, using dynamic requests generation in the database.

Let us make an assumption: a similarity between two skills is defined by the number of common descriptors among the skill internal attributes, the skill domain, the elementary competencies, the past projects, the evaluation of the skill in previous projects, the project domains, and finally the types of project domains. This information can automatically be obtained using the database table tree. Currently, the database structure analysis shows that, in this tree, the lower the depth of a table is, the more significant the similarity also is (see Figure 2). To find similarities, the S skill agent analyses the database table tree (the MAS environment), dynamically and recursively.

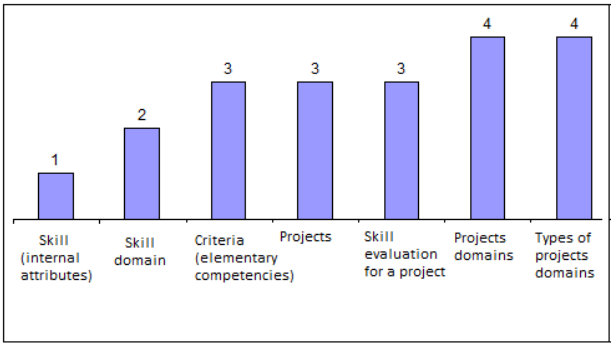

Figure 2. Similarities table depths

Let us assume that the maximum depth of the table tree is **D, which means that each skill can be reached from another skill after D links.** D=4 in the example of Figure 2, at the beginning of the tree exploration, the current table depth is **d=1**. The analysis starts with the global skill agents table.

Step 1: for each field of a current table, an SQL request is dynamically generated to find out the other skills sharing the same field value.

Step 2: if a skill S_i is found, the first time, its current proximity coefficient is $\mathbf{P_{Si} = 0}$.

Step 3: considering **n** as the number of common field values into the current table, P_{Si} is updated using equation (4).

$$P_{Si} = P_{Si} + (n \times (D - d)) \quad (4)$$

Step 4: Foreign keys and joined tables of the current table are used to define the next nodes of the tree. d is incremented: **d=d+1**.

Step 5: if **d** is not greater than D, go to step 3

Step 6: finally, all P_{Si} are normalized using equation (5):

$$P_{Si} = \frac{P_{Si}}{P_{Si\,\max}} \times 100 \quad (5)$$

The higher the proximity coefficient, the more skills are considered as potentially linked. A threshold is applied on Ps (e.g., 65% by default), and a list of potential linked agents is returned to the user for validation. An important point to know: validating the links implies the validation of a dedicated memory and an automated evolution of S agent behavioral rules. Thus, there is here again a real learning mechanism because of those evolutions arising from human validation. Reminder: when a skill agent is in the mature age, no human validation is needed and the agent is completely autonomous, generating links and updating its own behavioral rules.

F. Improving skill management using objectives

1) Introduction

"I want to improve the energy balance of my house by installing photovoltaic panels on the roof". "I want to win the rugby match next Sunday". Generalizing, the question is always to find the best answer to questions like "I want to do something to achieve something". Then the project comes into place. Goals and objectives are statements that describe what the project will accomplish. Each project is defined (structured) by a set of resources and processes according to a specific schedule. The processes are based on a set of required skills. Importantly, the definition of the project may evolve according to environmental conditions or past experiences, meaning that the list of skills required to reach the goal may not always be the same. Let us consider the goal of "building a house". Even if the process is almost the same, different construction materials (like cinderblock or brick) may be used. This implies different skills for each building project. Considering past experience in the domain, a dynamic dimension is observed for each project over time.

Then, the point is how to improve the management of skills to make them more efficient, throughout projects and over time. If a user wants to address a new goal, how should he define the project and what are the skills needed to implement it? How can one get benefits from past projects?

In order to solve the problem, work on the goals and skills of past projects is suggested.

2) Building new projects from past projects

Two cases are available to build the new project, by duplicating or customizing existing ones. From our previous example, we will suppose that the user finds the project "*integration of glass wool in walls*". He may then:

- Think that it is exactly his objective. Thus, he will duplicate this project and all its skills, without creating a new one, changing only contextual information like the start date for example.
- Think that it is not exactly his objective, but is very close. He decides to create a new project and customizes the list of skills associated with the existing project. A new objective is thus created with a new list of skills built from a subset of the previous one.

Finally, in both cases, a new project instance is generated from the objective. In our implementation, those operations are done throughout our unique *Objective* agent into the MAS. As those actions are based on historical data, no communication with skill agents themselves is needed. Let us assume that a new project has to be built according to the user needs. There is sometimes a limited expression of needs at the specification level and the objective might be met for the first time. The user would probably not know how to address the problem and how to exploit past projects. There is nevertheless a solution to help the user. The idea is to determine all the skills involved in past projects in the same objective domain, or eventually in the same environment. In our system, a skill "***wants to be involved***" in the new project according to a degree of involvement in past projects (see below). In contrast to what we have seen before, this method is not limited to an exchange with the *Objective* agent. In cognitive science, effective agents are obtained by the embodiment of mind [11, 12]. If a skill alone has no perception, no motivation and no means to perform an action and change its environment, it is always possible to define it artificially. Several types of motivations have been integrated in skill agents: contribution to new projects, determining the list of elementary competencies that define themselves, and determining their relationships with the other skills. Let us develop an example showing the motivation of being involved in new projects. A user defines a project in the "*thermal insulation improvement*" objective domain. No more detail is forwarded to the system. "*Thermal insulation improvement*" is an objective domain and is part of the environment "*sustainable building sector*". The answer from our system is defined by the following process:

- The *Objective agent* receives the user request.
- The *Objective* agent sends (broadcasts) the request to all skill agents within the MAS.
- Each skill agent computes a "*relevance coefficient*" according to the request content, and returns an answer to the Objective agent (see next paragraph).
- The *Objective* agent consolidates all answers from skill agents, and returns the list of candidates to the user

Each skill agent is autonomous and decides if it wants to contribute to the new project (or not). The key point is the computation of the relevance coefficient. Currently, this is the percentage of projects in which the skill has been involved in the past among all the projects of the objective domain. If it exceeds a threshold, the skill agent wants to be involved in the new project.

III. IMPLEMENTATION AND RESULTS

A. Implementation

1) The MAS Architecture

The model has been implemented using the JADE MAS and standard multi-agent tools [8, 9, 26, 27]; see Figure 3.

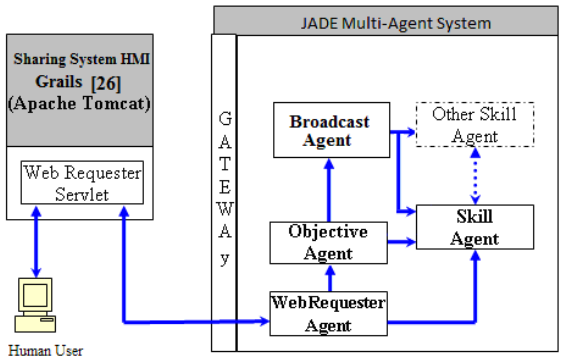

Figure 3. The MAS.

The JADE MAS has been integrated in standalone software, running into a Java Virtual Machine, and is called "SMAServer". The main components of the global architecture are:

- **User workstation**: exchange using a web browser
- **WebRequester Servlet**: This component is used for the management of the exchanges between human users and the MAS itself. It is a JADE MAS specific architecture component.
- **Gateway**: It is also a standard component of a JADE MAS, allowing dialogues among agents operating within the SMA and external programs (WebRequester Servlet) [8].
- **WebRequester Agent**: This agent is in charge of all the interactions with the human user. It forwards requests to skill agents and sends back their answers. It guarantees (FIPA compliance) that no direct exchange is possible between human users and skill agents.
- **Objective Agent**: According to Ferber's classification, the objective agent is reactive [3]. When a new project (the instance of an objective) is inserted within the system, information messages are

broadcast to all skill agents through the technical Broadcast agent one.

- **Broadcast Agent**: This is a technical (generic) agent. It receives an incoming message from a caller agent (e.g., Objective Agent), sends this message to all agents (e.g., skill agents) within the MAS and returns the answers to the caller.
- **Skill Agents**: actions have already been presented in subsection D.2. In our concrete implementation, they have sensors and effectors (as defined in standard agents theory [3]), each of them being a Java component (a class of object).

2) The user side Skill Sharing System HMI

The HMI from the user side has been implemented using the GRAILS [26] framework. There are 5 windows:

1. All skills that could be requested within the tool
2. All skills requests in progress
3. The connected user personal requests in progress
4. The skills shared (offered) by the connected user
5. The projects to which the connected user is involved. Here, the works in progress, within projects, for the connected user, are available.

TABLE I. LIST OF THE BEHAVIORAL RULES

XML Tag	Attribute	Mandatory	Comment
rules	description	X	Main tag
ruleGroup	description		Text describing the rule group
	weight		
rule	description	X	Text describing the rule
	weight		Weight of the rule in the rule group
	mandatory		Value is 1 if rule is mandatory, 0 otherwise
when	description		Text describing the condition
	sensor	X	Sensor Java class name used to verify rule condition
	params		Parameters in format name=value, separated by | character. Passed to the sensor
	result		Result variable name beginning with $
	operator	X	Logical operator used within condition expression
	table		Table name from which we try to verify condition
	field		Table field name or variable name from which we try to verify condition
	value		Value of field attribute, expressed as a regular expression
otherwise			Used if **<when>** has not been verified
do	effector	X	Effector Java class name to start if rule condition is verified
	description		Text describing the action
	params		Parameters in format name=value, separated by | character. Passed to the effector.
	result		Result variable name, beginning with $

3) Behavioral rules

The behavioral rules have been implemented in XML format with a specific grammar (hierarchy, attributes, and tags); see Table I. According to the agent links learning mechanisms, those rules may evolve over time if link creations are validated by the user. For illustration purposes, we propose below the rule ensuring that an agent will grow from childhood to teenage when 3 evaluations (e.g., after 3 concretizations across various projects) are available in the database:

```
<rule description="Growing from Youth to Teenage"
   mandatory="1" weight="1">
   <when description="Growing Conditions"
      sensor="GrowingSensor"
      params="unitaryEvaluationNumber=2"
      result="$evaluationsNumber1"
      operator="EQ" value="3">
      <do description="Grow to teenage"
         effector="Grow"
         params="from=youth|to=teenage" />
   </when>
</rule>
```

4) Skill agent memories

Each skill agent owns a dedicated memory table in which it stores the incoming parameters and the related computed decisions (see Table II).

TABLE II. MEMORY TABLE OF A SKILL

Field	Type	Comment
code	Long integer	Memory unique key code
ev_date	Date	Event date (record creation date)
evt_id	Long integer	Foreign key to event type table, describing the type of memorized event
agentid	Long integer	The current skill agent unique id
parametersin	String	Request parameters list in string format
decisionstring	String	Computed decision in string format
Humanvalidate	Boolean	Decision validated (or not) by human action
Comment	String	Free field

An example of memory record content is presented below. This result is obtained following the request for a list of linked skills to the skill with id 3192 (see Table III).

If the proposal is validated, the field *Humanvalidate* will then be set to "*true*" and a behavioral rule will be automatically generated.

5) The actor selection simulation tool

In Gironde, 61 of the local authorities are part of an "SD Network", where they share experiences and skills. They had started using and testing the system by the middle of the year 2013. The experiment concerned the management of SD projects. A preliminary study has been carried out, showing that most projects fall into 9 domains of activity.

TABLE III. MEMORY RECORD EXAMPLE

Field	Type	Comment
code	Long integer	1
ev_date	Date	2013-09-10 14:38:55.345+02
evt_id	Long integer	4
agentid	Long integer	3192
parametersin	String	controller=RDEngine;stepNumber=1;signalCode=201;action=callRdEngineForAgentLinks;fromWebInterface=true;minPercentValue=1;agentId=1508
decisionstring	String	SELECT agent.* FROM agent, agent_agt_domain, agt_domain WHERE text(agt_domain.code)=text(agent_agt_domain.agt_domain_id) AND text(agent_agt_domain.agent_domaines_lies_id)=text(agent.code) AND text(agt_domain.descriptiondomaineagt)=text('3-Diagnostic') AND agent.code<>1508
Humanvalidate	Boolean	False
Comment	String	Find agent links

These domains are: political wishes, sensitization, diagnostic, prospective, developing the strategy, elaborating the action plan, implementation of the action plan, evaluation, and continuous improvement. Skills are related to one or more domains. For instance:

- The skill “animation capability” is attached to the “political wishes” and “diagnostic” domains.
- The skill “identification and mobilization of expertise” is attached to the “prospective” domain.
- The skill “development of the sustainability report” is attached to the “continuous improvement” domain.

For skill actors selection, in order to obtain immediate results, a simulator was built to verify the theoretical proposal. It is based on elementary competencies weight computation. The simulation phase lies in a call to a unique skill agent, with the aim of observing its behavioral evolution over time. Each simulation applies random values to weights to each elementary competency (4 in total) of this skill. During the simulation phase, 100, 200 and 300 requests and validations were done.

B. Results

1) A real life experiment at CG33

It is observed that the skills evolve in a “real life” SD Skill Sharing System, at CG33, and they provide answers. The following results are presented in this context. The positive point is that the skills provide valuable information to the actors who have poor understanding of the elementary competencies. The drawback, however, is that the initialization of the system is labor intensive. The first definition of the skills requires strong expertise in the domain. The updates can be done at any time, but it takes a long time to collaborate with experts in order to capitalize their knowledge and insert relevant skills and elementary competencies into the system. Therefore, it is difficult at the moment to conclude about the efficiency of our model because we are still in the early stages of the tests. We hope to present interesting results in the near future.

In order to demonstrate the versatility of our proposal, other tests have been performed using another functional domain: the selection of the best players for rugby. In this application, each player's position is considered a different skill. Elementary competencies are for instance the ability to tackle and stop an opponent or to be accurate in kicking the ball. The evaluation of a player for the embodiment of a given skill is based on his performance for each criterion and on the number of selections. When the system is asked to suggest a player for a given skill, equation (3) is used. Then the propositions elaborated by each skill agent are validated (or not) by the user, the players are evaluated and the database is updated. The results are positive for the identification of players over the different iterations.

2) Actors selection simulator

Interesting results were found using the simulation tool to verify our hypothesis. The dynamic computation, with memorization, for weights of criteria is valid and convergent over time. An example of a convergence graph (here for 2 criteria over 4), across simulations, is shown in Figure 4. In this example, the weight of the first criterion is converging to 0.8, and the weight of the second one is converging to 0.6.

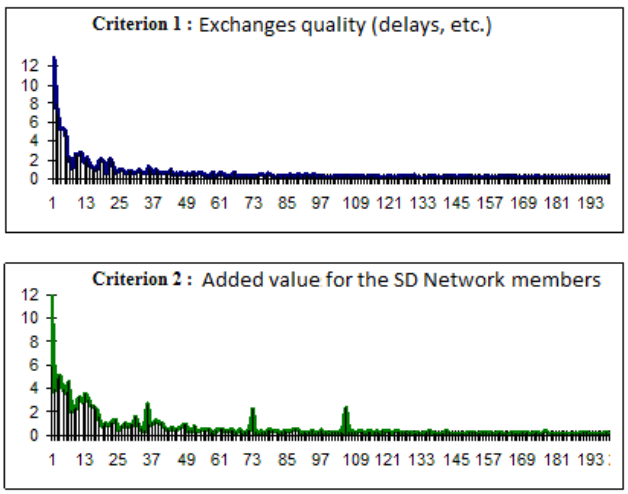

Figure 4. Convergence of criteria values

It has also been found that, for a given skill agent, the learning mechanism is efficient and can be considered as becoming “stable” when the system has stored around 120 requests and validations coming in from users. Another interesting point is the comparison of the values of criteria weight across simulations (see Figure 5).

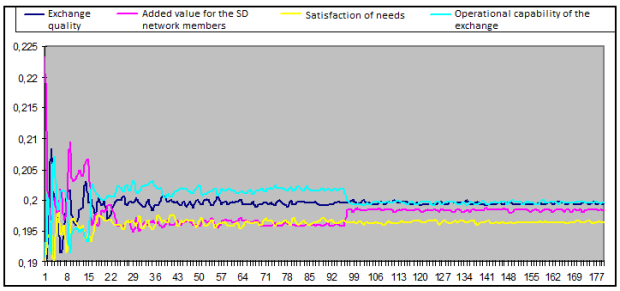

Figure 5. Compared criteria weights values

It does not seem meaningless to admit, from a theoretical point of view, that an elementary competency E_1 is "more important" than another E_2 for the global skill definition if the weight of E_1 is higher than the weight of E_2. If the graph above shows this fact, outside of a simulation process, in "real life" conditions, this observation will allow us to identify the most important elementary skills for a given skill definition. Another conclusion is that our proposal is versatile, applicable to any professional activity domain, and not only to SD projects. The example of the selection of players for rugby positions, a very different domain, is also possible as mentioned before in this article.

3) New project creations

a) Case 1: new projects from past objectives

The user request, through the *WebRequesterAgent*, is transmitted to the *Objective* agent that processes it (see Figure 6).

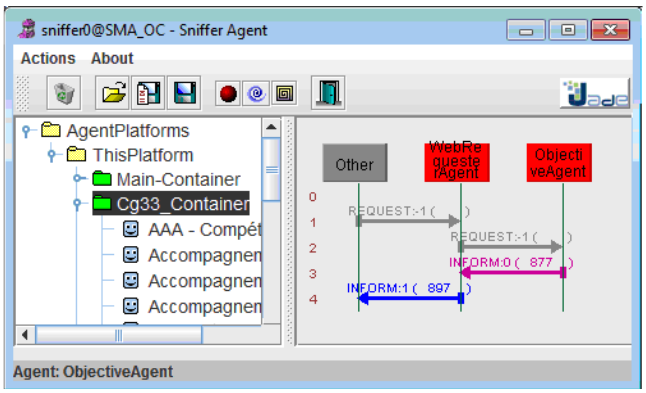

Figure 6. Case 1 - Exchanges between agents into the JADE MAS

Figure 6 is a screen copy of the JADE MAS Sniffer tool that monitors message exchanges between agents. The left part shows the MAS Agent tree where an agent is part of a container, a container part of a platform (*ThisPlatform*) and a platform part of all agent platforms (*AgentPlatForms*). The right part shows three "boxes":

- *Other*: reflects other agents within the MAS, or in this case, the JADE gateway that manages exchanges with the external users
- *WebRequesterAgent*: manages the interactions with the human user, forwarding requests to other agents, and sending back their answers.
- *ObjectiveAgent*: see next paragraph for details.

The arrows (1 to 4) show the message exchanges, with their type (REQUEST or INFORM for the answer), and their directions (from sender to receiver).

The *Objective* agent ensures the treatments, based on historical data, in retrieving project instances and related skills. At the end, it processes the answer in the form of an XML flow (see Table IV).

TABLE IV. XML FIELDS INTO THE ANSWER FLOW

XML Tag	Comment
answers	Main tag encapsulating the answers
answer	Main tag for each answer, 1 for each project
newObjective	Boolean value indicating that the project is new. Value always false here because the project is over and taken into the historical
objective	Main tag for the past project
code	The past project code within the projects database table
description	The past project textual description within the projects database table
startDate	The past project start date
endDate	The past project end date
skills	Main tag encapsulating the skills list
skill	Main tag, 1 per skill
code	The current skill code within the skills database table
description	The current skill textual description within the database table

At the front-office user level, a list of projects and skills is proposed. The new user project is then generated according to one of the two methods identified: duplicate or customize.

a) Case 2: new projects from objective domain

The user request is sent to the Objective agent that does a broadcast to all skill agents and consolidates their answers. At the end, the answer is also returned to the user as an XML flow, as already shown in case 1. The "BroadcastAgent" is a technical one and reusable by all other agents if they need to do such "broadcasting" actions in the future. See Figure 7 for a screen copy of the JADE MAS Sniffer tool. Only the right part of the window is shown for better visibility. The three first "boxes" are identical to those described for Figure 6. Please note that BroadcastAgent receives a message from an original sender, broadcasts it to all agents, and consolidates all answers into a single one. The global answer, an XML flow, is then sent back to the original sender. All other boxes on the right are all skill agents within the MAS. The screen copy proposed Figure 7 only shows two skill agents.

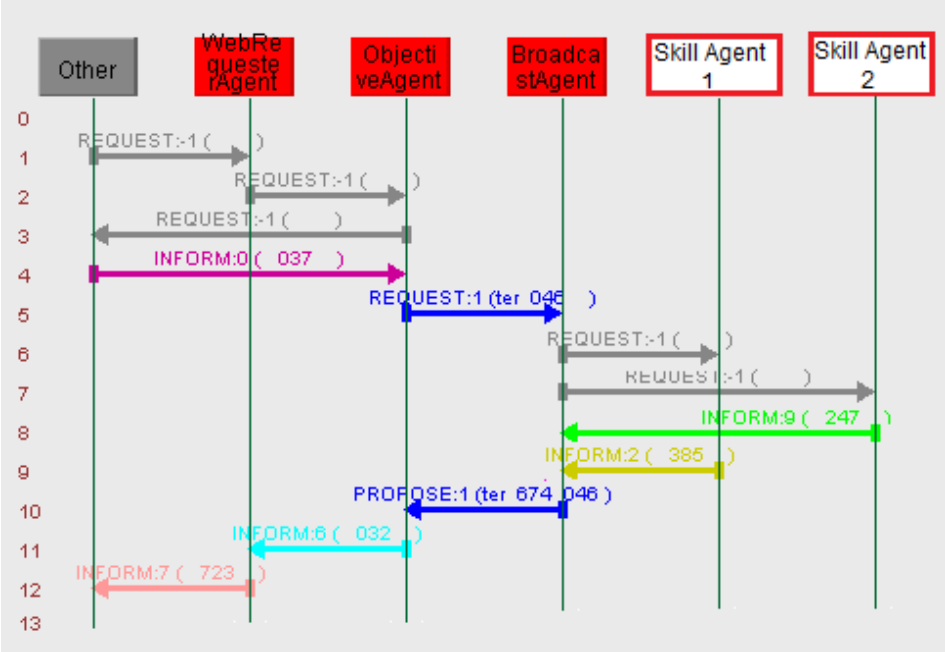

Figure 7. Case 2 - Exchanges between agents into the JADE MAS

For illustration, let us reuse the example where the objective domain is "*new means of energy production*". A project for a "*wind turbine implementation*" has five phases:

1. *Project management / implementation*
2. *Definition of power requirements*
3. *Selection of the best wind turbine technology among models and worldwide suppliers*
4. *Logistic definition (transportation organization)*
5. *Wind turbine installation*

A second project, entitled "*hydraulic micro power implementation*", shares 3 (over 5) common phases with the first project, and has three *specific ones*:

1. *Project management / implementation*
2. *Definition of power requirements*
3. *Selection of the best hydraulic micro power technology among models and worldwide suppliers*
4. *Logistic definition (transportation organization)*
5. *Hydraulic micro power installation*
6. *Technology transfer of appropriate designs to developing country manufacturers*
7. *Project formulation and appraisal for national and international aid agencies*
8. *Training on small-hydro technology and economics*

A third one is entitled: "*installation of a biodiesel generation system to power up a highway construction site*". Let us assume the project also shares the 3 common phases, and has *two specific ones*:

1. *Identify the best site*
2. *Environmental benefits evaluations*
3. *Project management / implementation*
4. *Definition of power requirements*
5. *Selection of the best technology of biodiesel generation system among models and worldwide suppliers*
6. *Logistic definition (transportation organization)*
7. *Biodiesel generation system installation*

Considering these phases as skills (of course at a high level of abstraction), let us introduce into the system a new user request where the new project is "*solar panels implementation*". This project also belongs to the objective domain "*new means of energy production*". The requested minimum value for the relevance coefficient is 75%. Thus, according to our algorithm:

- The *ObjectiveAgent* sends (broadcasts) the user request to all the skill agents within the MAS (here 3 common + 5 specifics according to the second and third project phases)
- Each skill agent computes its dedicated relevance coefficient. If this computed value is greater than the requested one, that means the skill agent "wants" to participate in the new objective, and the boolean value "*true*" is returned to the *ObjectiveAgent*
- The *ObjectiveAgent* consolidates the results where the answer is "*true*". Finally, it returns to the user the list of the skills as an XML flow:
  a - Project management / implementation
  b - Definition of power requirements
  c - Logistic definition (transportation organization)

- Through the front-office interface, the user then validates (partially or totally) the skills list to build its new project. This validation is stored in the memory.

C. Discussion

1) Skill Agents

In most SMA applications, the skills are not agents. They are typically described by behavioral rules that determine the actions of the agents [28]. The difficulty is often in making the link between tacit and explicit knowledge and learning from the real world [27, 28]. For instance, in other applications such as the management of skills in the context of e-learning, one of the main problems is to determine and make explicit the tacit knowledge that has not been understood and to adapt the courses [29]. The advantage of our approach is that it is skill centered. The skills are learning agents and their motivation is to determine the list of elementary competencies that define themselves and their relationships with the other skills. These elementary competencies usually correspond to tacit knowledge and know-hows that cannot be easily defined. One of the key ideas of our model is to consider that the weighted list of criteria defined by the users to determine the best actor for a given skill are abstractions of a hidden list of elementary competencies. The system learns from the requests of the users.

2) Skills management improvement using objectives

The proposed information system also offers an answer to the problems of each SD Network Member at CG33: they need to identify the required skills to make their project successful. Whatever the activity sector, project management is usually done through software applications, where tasks are defined and described in a static way. The definition of a new project requires the identification of human actor(s) for each of them. The first problem occurs when there is an evolution of the processes. A traditional approach is to update statically the list of tasks for the project. Settings renewals have to be done by administrators or advanced users in the project management tool itself. This update is often a generator of costs, because in some cases an external help (e.g., by the software editor) may be required. In this approach there is a lack of efficiency, inducing at least a waste of time, and sometimes some additional financial charges that could be substantial. Another problem is the management of projects dynamics according to the user needs at a given date. As a project reflects the user needs at this date, there can be as many projects as user needs expressions within the system. The global skill sharing system presented in this paper is a collaborative tool and not a static one. The new projects are built “on the fly”, from the real user needs. Thus, the new skills list is built from those available in past projects and reflects the user needs at the time of the request. As the number of projects grows in our collaborative system, the global list of skills and the objective domains evolve and may converge. At a global level, our system learns from the requests of the users and reflects the evolution of activity over time. The observation of those evolutions will drive the management of the company to put the focus on certain skills or to drop them. Our proposal therefore provides an interesting answer to the problem of skills management, using objectives, at the project and organization levels.

IV. CONCLUSION AND FUTURE WORK

A multi-agent system has been proposed for skills sharing between actors in collaborative projects in the domain of sustainable development. The key point of this work is the definition of skills as agents with their own rules for learning and evolving in an environment where actors are considered as resources for the embodiment [12] of the agents. If it is always necessary to use skills within projects, the choice of the human actor to implement them is not "suggestive and human centered" anymore. Over time, more and more objectives will be concretized through projects, and more and more information will be available to help the user. This work suggests interesting perspectives. From a professional point of view, concerning the problem of skills management, the analysis of skills applications through projects provides a wealth of information. After a period of running a project, managers and human resources management services will be able to identify the key skills, the cross-domain skills and the evolution of the “sensibility” of each skill in the professional sector processes, and all of this over time. This work may introduce real benefits into human resources management to anticipate future evolutions of needs in terms of collaborators’ profiles. Whatever the professional activity domain, the skill sharing system ensures reciprocity, human cooperation and versatility. Going further, considering that some skills may be viewed as "cross-domain" and/or "cross-companies", what about impacts on the future organization of work ?

Several issues have been identified for future works. The current tests have to be deeply validated in “real life” environment. The system has to be tested with a comprehensive list of skills (currently 110 at CG33) and elementary competencies provided by experts of the domain. A large number of evaluations are also required to test the evolution of the agents at different ages. Talking about skill management improvement, the dynamism over time is introduced by the computation of a pertinence coefficient value. At this time, this value is currently stored in the memory of each skill agent but not used in future iterations. More work has to be done to study the computation optimization of this value, and its integration in the learning mechanisms.

When talking about skill management improvement using objectives, the adaptation over time is made possible by the computation of a relevance coefficient value. This value is stored in the memory of each skill agent. One of the future direction for works is implementing complementary

learning mechanisms to optimize the computation of this value over time.

REFERENCES

[1] O. Chator, J.M. Salotti, and P.A. Favier, “Multi-agent system for skills sharing in sustainable development projects,” Proceedings of COGNITIVE 2013, the 5th International Conference on Advanced Cognitive Technologies and Applications, pp. 21-26, IARIA Conference, Valencia, Spain, 2013

[2] M. Bardou, “De la stratégie à l’évaluation : des clés pour réussir un Agenda 21 local,” (“From the strategy to the assessment: the keys to succeed in the elaboration of a local Agenda 21”) , Collection “Références” du Service de l’Économie, de l’Évaluation et de l’Intégration du Commissariat Général au Développement Durable, Paris, France, 2011

[3] J. Ferber, “Multi-Agent systems. An introduction to distributed artificial intelligence“, Addison Wesley, London, UK, 1999

[4] N.R. Jennings, M. Wooldridge, and K. Sycara, “A roadmap of agent research and development,” Journal of Autonomous Agents and Multi-Agent Systems, vol. 1, 1, pp. 7-38, Boston, USA, 1998

[5] G. Weiss, ”Multiagent systems, a modern approach to distributed artificial intelligence,” MIT Press, USA, 2013

[6] ADEME, “Bâtiment et démarche HQE,” ("HQE building and methodology"), Collection Connaître pour agir, Paris, France, 2006

[7] S. Belkada, A. I. Cristea, T. Okamoto, ”Measuring knowledge transfer skills by using constrained-student modeler autonomous agent,” Proceedings of the IEEE International Conference on Advanced Learning Technologies, IEEE Computer Society, 2001

[8] FIPA, “Foundation for intelligent physical agents - Abstract architecture specification (standard version),” publication of FIPA’s Technical Committee, Geneva, Switzerland, 2002

[9] F. Bellifemine, A. Poggi, and G. Rimassa, “JADE – A FIPA-compliant agent framework,” 4th International Conference on Practical Application of Intelligent Agents and Multi-Agent Technology, PAAM-99, pp. 97-108, London, UK, 1999

[10] R. Pfeifer and J.C. Bongard, ”How the body shapes the way we think: a new view of intelligence,” The MIT Press, USA, 2007

[11] L. Shapiro, “Embodied cognition,” in Oxford Handbook of Philosophy and Cognitive Science, E. Margolis, R. Samuels, and S. Stich (eds.), Oxford University Press, UK, 2010

[12] F. Varela, E. Thompson, and E. Rosch, “The embodied mind: cognitive science and human experience,” MIT Press, USA, 1991

[13] M. Mulder, T. Weigel, and K. Collins, “The concept of competence in the development of vocational education and training in selected EU member states: a critical analysis,” Journal of Vocational Education and Training , vol. 59, 1, pp. 67-88, 2007

[14] M. L. Putterman, “Markov decision proceses. Discrete stochastic dynamic programming,” Wieley-Interscience, New York, USA, 1994

[15] M. Mitchell, “Machine learning,” chap. 13 “Reinforcement learning,” pp. 367-390, Ed. McGraw-Hill Higher Education, USA, 1997

[16] O. Von Neumann and O. Morgenstern, “Theory of games and economic behaviour,” Princeton University Press, USA, 1944

[17] L. Shapley, “Stochastic games,” Proceedings of National Academy of Science, 39, pp. 1095–1100, USA, 1953

[18] C. Boutilier, T. Dean, and S. Hanks, “Decision theoretic planning: structural assumptions and computational leverage,” Journal of Artificial Intelligence Research, USA, 1999

[19] E. Plaza and S. Ontañon, “Cooperative multi-agent learning in Adaptive Agents and Multi-Agent Systems,” Lecture Notes in Computer Science, vol. 2636, pp. 1-17, Spain, 2003

[20] D. McSherry, “Completeness criteria for retrieval in recommender systems,” Advances in Case-Based Reasoning, 8th European Conference (ECCBR’06), pp. 9-29, Turkey, 2006

[21] H. Mühlenbein, “How Genetic Algorithms Really Work: 1. Mutation And Hill Climbing,” Manner, R. and Manderick, B. (eds), Proceedings of the Second Conference on Parallel Problem Solving from Nature, Elsevier Science, vol. 2, pp. 15-25, Belgium, 1992

[22] S.C. Shiu, K. Shiu, D.S. Yeung, C.H. Sun, and X. Z. Wang, “Transferring case knowledge to adaptation knowledge : An approach for case-base maintenance,” Computational Intelligence", vol. 17, issue 2, P 295-314., 2001

[23] J.R. Quinlan, “C4.5 : Programs for Machine Learning,” Proceedings of European Conference on Machine Learning, pp. 3-20, Austria, 1993

[24] P. Domingos, “Unifying instance-based and rule-based induction,” Machine Learning, vol. 24, issue 2, pp. 141-168, 1996

[25] M.J. Zaki, C.J. Hsiao, “CHARM : An efficient algorithm for closed itemset mining,” Proceedings of the Second SIAM International Conference on Data Mining, pp. 12-28, USA, 2000

[26] G. Rocher, “The definitive guide to Grails,“ Apress, New York, USA, 2006

[27] S. Goderis, “On the separation of user interface concerns: a programmer's perspective on the modularisation of user interface code,” Ph.D. thesis, VrijeUniversiteit Brussels, Belgium, 2007

[28] H. Friedrich, O. Rogalla and R. Dillmann, ”Integrating skills into multi-agent systems,” Journal of Intelligent Manufacturing, vol. 9, 2, pp. 119-127, Karlsruhe, Germany, 1998

[29] A. Garro, L. Palopoli, ”An XML multi-agent system for e-Learning and skill management,”, proceeding of: Agent Technologies, Infrastructures, Tools, and Applications for E-Services, NODe 2002 Agent-Related Workshops, Erfurt, Germany, 2002