LOUVAIN
School of Management
RESEARCH INSTITUTE

WORKING PAPER

2016/17

Designing a MOOC as an Agent-Platform Aggregating Heterogeneous Virtual Learning Environments

Yves Wautelet,
KU Leuven, Belgium

Samedi Heng,
Louvain School of Management Research Institute

Manuel Kolp,
Louvain School of Management Research Institute

Loris Penserini,
Istituo Tecnico Commerciale “Cesare Battiti” , Italy

Stephan Poelmans,
KU Leuven, Belgium

Louvain School of Management Research Institute
Working Paper Series
Editor : Prof. Frank Janssen
(president-ilsm@uclouvain.be)

Designing a MOOC as an Agent-Platform Aggregating Heterogeneous Virtual Learning Environments

Yves Wautelet, KU Leuven, Belgium
Samedi Heng, Louvain School of Management Research Institute
Manuel Kolp, Louvain School of Management Research Institute
Loris Penserini, Istituto Tecnico Commerciale "Cesare Battisti", Italy Stephan Poelmans, KU Leuven, Belgium

Summary

With the emergence of cloud technologies on the one hand and social networks on the other hand, the possibilities for e-learning have been considerably enhanced in the latest years. Virtual Learning Environments (VLE) can now indeed contain a huge amount of learning resources; in parallel large user communities are available in social networks. Nevertheless, these systems remain different but, by using these heterogeneous software environments together, the possibilities for interaction could be multiplied. That is why, this paper suggests building a Massive Open Online Course (MOOC) environment through a Multi-Agent System (MAS) working as a virtual abstraction layer over heterogeneous software platforms. The idea is to aggregate different traditional VLE to dispose of the learning objects they own as well as other platforms like social networks to furnish an easy access to the MOOC of their large user communities. The MAS design and implementation has been architectured around a real-life organizational pattern – the joint venture – allowing us to deal with the complexity of heterogeneous software environments in the fashion that real-life companies join governance and management. Communication scenarios issued from a field analysis are pointed out in the paper; these are supported by the MOOC platform in the native environment as well as in Facebook. The proposal is indeed validated through the development of a prototype using Facebook as a case study for third-party platform interfacing. We finally highlight the benefits for the user experience.

Keywords : Agent Architecture, Agile Development, User Story, Agile Architecture, Multi-Agent System.

This paper is accepted for publishing in the special issue of Behaviour & Information Technology Journal 2016.

Designing a MOOC as an Agent-Platform Aggregating Heterogeneous Virtual Learning Environments

Yves Wautelet[1], Samedi Heng[2], Manuel Kolp[1] Loris Penserini[3], and Stephan Poelmans[1]

[1] KU Leuven, Belgium
{yves.wautelet, stephan.poelmans}@kuleuven.be,
[2] Universite catholique de Louvain, Belgium´
{samedi.heng, manuel.kolp}@uclouvain.be
[3] Istituto Tecnico Commerciale "Cesare Battisti", Italy
elpense@gmail.com

Abstract. With the emergence of cloud technologies on the one hand and social networks on the other hand, the possibilities for e-learning have been considerably enhanced in the latest years. Virtual Learning Environments (VLE) can now indeed contain a huge amount of learning resources; in parallel large user communities are available in social networks. Nevertheless, these systems remain different but, by using these heterogeneous software environments together, the possibilities for interaction could be multiplied. That is why, this paper suggests building a Massive Open Online Course (MOOC) environment through a Multi-Agent System (MAS) working as a virtual abstraction layer over heterogeneous software platforms. The idea is to aggregate different traditional VLE to dispose of the learning objects they own as well as other platforms like social networks to furnish an easy access to the MOOC of their large user communities. The MAS design and implementation has been architectured around a real-life organizational pattern the joint venture allowing us to deal with the complexity of heterogeneous software environments in the fashion that real-life companies join governance and management. Communication scenarios issued from a field analysis are pointed out in the paper; these are supported by the MOOC platform in the native environment as well as in Facebook. The proposal is indeed validated through the development of a prototype using Facebook as a case study for third-party platform interfacing. We finally highlight the benefits for the user experience.

Keywords: Multi-Agent Systems; Organizational Patterns; Social Patterns; Joint Venture.

1 Introduction

The traditional classroom, equipped with chair, slate blackboard and benches arranged in rows, has been progressively augmented by various technologies, which have become the third element of the interaction between teaching staff and students. Today the most advanced classrooms go beyond this logic and also include the use of mobile

devices and digital whiteboards that enrich the traditional learning approach by additional learning channels more familiar to nowadays students, called “digital natives”. Thanks to the Web 2.0 technologies, which includes Internet tools or applications for social media and social networking, users can create their own interactive contents and make them available to other users. The Web 2.0 offers instruments to move a step forward with respect to traditional *Virtual Learning Environments* (*VLE*, see [50,12]), which are considered a one-dimensional interaction approach as instructors would post lecture notes on those systems via a link or click. Therefore, integrating traditional and Web 2.0 technologies, educators can move towards a new generation of classroom, the Classroom 3.0 – as coined by [15] – that should have no physical boundaries and will be accessible anytime and anywhere.

A lot of new platforms like *Facebook* and *LinkedIn* were developed for social networking; those were not immediately designed for learning purposes (or as VLE) but contain huge amounts of users immediately able to communicate to each other and potentially interested in learning activities. An integration of these platforms with one or several VLE could thus enable us to dispose of huge user communities without constraint of having to log in a different software environment, which in many ways positively impact the user experience (see Section 2 for a detailed positioning on this question). Challenges in VLE thus include furnishing tools allowing more and more user groups of already existing software platforms to actively communicate with a reasonable time to answer and efficiently share knowledge in heterogeneous software environments. Actors involved in the learning process are then likely to use different supporting software systems. Heterogeneity could be supported by setting up a software environment designed as an upper layer to classical VLE like *Learning Management Systems* (*LMS*), *Content Management Systems* (*CMS*), *Higher Education and Research Solution Portfolios for ERP systems* but also other software systems disposing of large user communities like social networks. The use of this upper-layer system can then be used to ensure efficient communication between the user groups and roles leading to build a *Massive Open Online Course* (*MOOC*).

A MOOC is an online course aimed at unlimited participation and open access via the web (see [52,11]). In addition to traditional course materials such as filmed lectures, readings, and problem sets, many MOOCs provide interactive user forums to support community interactions between different peers like students, professors and teaching assistants. Communication canals and abilities need to be supported and managed adequately in order to ensure that requests of learning peers are addressed correctly and in time but without waste of time for instructors (and thus waste of money for the Educational Institution). Indeed, when thousands of users subscribed and actively follow the MOOC asking questions online, it is impossible for a single instructor to reply to them all. Most of the questions can be answered by other learning peers and only a small number should effectively be escalated to instructors. Instructors thus only need to effectively answer a small fraction of questions and, in most of the cases, only validate answers of third-party learning peers to keep the

system manageable but also valid. Properly managing this communication process in an easy-to-access software environment is vital to keep the motivation of the learners high (avoiding that they massively abandon the course).

VLE at large and MOOC in particular are by nature human and socially based structures [29]. Indeed, such systems require a set of (human) actors playing different roles and collaborating to achieve common and individual goals. The more actors involved in the learning process, the more content is produced and the higher the learning potential. Nevertheless this also increase its management and communication complexity. In order to better deal with this complexity we suggest, in this paper, to architecture a software platform as a *Multi-Agent System* (*MAS*). This MAS follows an organizational pattern (called the *joint venture*) in the analysis stage; very concretely, the advantage of applying patterns to the software architecture is to build software that maps its behavior on the behavior of individuals organizing themselves in order to learn, teach, communicate, etc (see Section 3).

The main contributions of the paper are:

- to build real-life communication scenarios/issues that can be found in MOOCs. Literature shows that communication is one of the key success factors for a MOOC (see Section 2);
- to architecture the realization of these scenarios around a collaborative MAS-platform (following organizational and social patterns). The collaborative platform can be said to be socially engineered (see Section 2);
- to validate the architecture through the development of a platform prototype that can, at run-time, behave in line with the users' behavior in order to synchronize communication content over multiple software environments like social networks thus acting as an upper layer over classical VLE. A MOOC that is accessible in a software environment the user is already familiar with positively impact its user experience (see Section 2). The development of the platform prototype serves as a proof-of-concept. In the proof-of-concept, Facebook is used as case study for third-party platform interfacing.

The goal of the paper is not to show that Facebook can be used as a VLE but rather to show that complex communication scenarios of a defined MOOC can be synchronized with third-party platforms like Facebook with the use of a MAS architecture for an easy access of their large user communities. The contribution belongs as much to computer as to social sciences. Indeed, besides the technical approach, we show that we can create software with a behavior aligned to the human organizational one in order to provide a solution to known communication issues found in e-learning and VLE. The impact on the user experience is immediate because the user can follow the MOOC using its existing software environment and does not imperatively require to log in a third-party environment; this is especially important since only about 10% of the users subscribed to the MOOC effectively keep their motivation to actively follow the lessons (so continue learning) until the end [22].

The paper is structured as follows. Section 2 depicts related work while Section 3 depicts the context and the method of the present research. The latter section indeed describes the i* framework used to model human and system behavior and generically explains the joint venture organizational style. Section 4 describes the (requirements) scenarios that must be supported by the proposed MAS architecture thus software platform. Section 5 then instanciates the generic joint venture organizational style to a MOOC behavior and describes the main roles involved in it. Section 6 describes the design of the MAS to be set-up and its behavior at run-time to support the defined scenarios. Section 7 describes the implementation of the MAS and illustrates its use through its interfaces. Finally, Section 8 concludes the paper.

2 Related Work

As overseen in the introduction, the contribution of this paper is located at the frontier of multiple research domains that we need to discuss in order to position it. That is why the contribution is positioned with respect to the evolution of computer science, social-engineering and the user experience.

2.1 Positioning with Respect to Computer Science

With the rise of cloud computing technologies (see [1] for a definition and characterization), the possible use of VLE as distributed architectures aggregating heterogeneous software systems to support learning processes has been identified. Indeed, [31] suggest this approach but keep their proposal limited and do not point to technologies and development paradigms to support it nor concrete learning scenarios. Their approach can be considered as the embryo of ours; we go further by showing how it can be implemented through communication scenarios using various technologies.

[24] identify that agents and MAS have been used for the two past decades in the context of e-learning in two ways: *(i) agents as a modeling and design paradigm for advanced human-computer interaction and (ii) agents for smart functional decomposition of complex systems*. In the same book, [46] nevertheless only develop the first aspect through the development of ICTEM, an inter-university collaboration VLE allowing building a virtual learning community based on a common course pool. Their VLE requires a common inter-university taxonomy for information classification and organization that all members of the virtual learning community agree upon. The common taxonomy allows building the course pool in order to share courses between providers (i.e. partner institutions (universities) providing teachers) and consumers (i.e. partner institutions providing students). A technique for keeping such a taxonomy (or ontology) up-to-date is seen in [18]. The taxonomy constitutes, in our opinion, the most interesting contribution of the paper because it allows the dialogue and sharing of resources. We use a comparable taxonomy, notably at the level of the so-called *Learning Objects* (see Section 4 for a definition), but we abstract their approach using

an agent-architecture acting as an integration platform for heterogeneous VLE. As said, [46] only consider agents as real-life collaborating peers (so approach (i)) while we build and map this real life organization to the design of the software (combining approaches (i) and (ii)). We thus consider that our work further develops their ideas by building a virtual agent community offering more integration possibilities than the combination of VLE of different universities only.

2.2 Positioning with Respect to Social Engineering

At second, we position our contribution with respect to the joint evolution of computer and social sciences.

Next to technology-based approaches, we aim to map in this paper the social behavior of individual agents with the system behavior at run-time. This trend is known, in literature, as social engineering [54]. Within this discipline, patterns of organizational and social behavior have been distinguished over the years. For instance, the joint venture pattern was firstly applied to a MOOC architecture by [27] but with no realization scenarios nor implementation. This paper addresses these issues by describing scenarios supported by the MOOC architecture and showing their concrete implementation. All in all, the approach allows us to dispose of a software system behaving following an experienced human organizational pattern to support processes involving a huge set of collaborating software agents. [19] emphasizes the necessity to understand the social and cultural aspects of software analysis and design when modeling applications involving a lot of users out of various contexts; they notably illustrate it with the modeling of a VLE.

So far in this section we have discussed on the one hand the possibilities brought by cloud environments and on the other hand the use of (socially engineered) MAS. Their first combined coverage has been done in [40]. The latter indeed identifies a paradigm shift from traditional e-learning systems to cloud based ones and highlights cloud-based VLE systems as suitable for an application in the context of agent-based software engineering. Indeed, the authors identify 5 key issues addressed by agents in the field of cloud e-learning systems in order to build a MOOC, i.e. *learner-centered*, *openness*, *personalization*, *self-motivation* and *collaboration*. These issues are also supported in various levels by the software platform developed here as a proof-of-concept of the contributions.

2.3 Positioning with Respect to User Experience

Let us finally tackle the problem through a user-oriented view. Facebook is often used to post links to elements of a MOOC present in the MOOC platform or communicate about the MOOC (e.g. [7]). Nevertheless, it is used in an asynchronous manner meaning that there is no auto update between Facebook and the MOOC native platform so that (unless very strictly managed) environments turn, in time, to be completely inconsistent. [47] suggest to exclusively use Facebook as a VLE. They

distinguish six base functions that need to be fulfilled in order to use the social platform as a VLE. The aim of the contribution in this paper is not to fully support each function of a VLE (or MOOC) through an agent-architecture but rather to support the complex communication issues found in MOOCs. Although already partially achieved, the study of the support of all possible MOOC functions with our platform is out of the scope of this paper that focuses on communication aspects and is left for future work.

As mentioned in the introduction, the proposed approach has an impact on user experience. Sets of papers identify the issues of proper communication and support in MOOCs. [17] showed that students who actively participate to discussions tend to better succeed in evaluations so that communication should be monitored and managed optimally. [22] explains that *Up to 90%* (of the students) *drop out due to reasons including* (S1) *a lack of incentive,* (S2) *failure to understand the content material and having no one to turn to for help, and* (S3) *having other priorities to fulfill*. In the same way, these authors highlight 4 key future challenges for instructors, i.e. (I1) *difficulty in evaluating students work,* (I2) *having a sense of speaking into a vacuum due to the absence of student immediate feedback,* (I3) *being burdened by the heavy demands of time and money, and* (I4) *encountering a lack of student participation in online forums*. All in all, most of the issues (S2, I2, I3 and I4) are impacted by our approach because they are related with communication.

Direct benefits of the MAS-based platform for students:

- with respect to (S2), an enhanced community to turn to. Various technical elements (e.g. immediate notifications, the #hash-tag function, ...) allow to browse requests for learning objects within the connected social network (e.g. Facebook) in a platform where users are more logged in than in a specialized MOOC one.

Direct benefits of the MAS-based platform for instructors:

- with respect to (I2), a more direct feedback through textual messages possibly using the Facebook *@mentioning* function, the *like* function, etc;
- (with respect to I3), an optimization of their answer time (see Section 4).

Indirect benefits of the MAS-based platform for instructors:

- with respect to (I4), mutual feedback through diverse communication means increases the overall conversations through a snowball effect;
- with respect to (I1) optimization of time devoted to the MOOC will give more time to work on activities like students evaluation.

3 Research Context and Method

In order to deal with the complexity of the research, the development of the platform is divided into 3 main stages: (i) the description of the requirements scenarios, (ii) the

MAS architectural design and detailed design to support these requirements and, finally, (iii) the implementation of the MAS through technical languages and technologies.

The understanding of the artifacts presented and realized during these stages do not require the knowledge of the full software engineering methodology nor technical aspects of (agent-oriented) software development. Nevertheless, the developments presented in this paper require the understanding of *i** models (see [54]) which offer actors (agents, roles or positions), goals, and actor dependencies as primitive concepts for modeling an application during early requirements or organizational/social analysis. Figure 1 distinguishes the i* models' elements and their graphical representation used in several diagrams of the paper. The models can be generically used in any software engineering method for modeling the organizational setting producing an architectural model of the future software system. We present the architectural solutions that have been chosen during the analysis and design of the software system and briefly overview their implementation.

The research has started with the identification of the issue of integrating heterogeneous VLE and the communication (requests for learning objects) scenarios that must be supported. These scenarios have been built using a field analysis leading to a precise identification of requirements (see Section 4). To build this heterogeneous VLE able to

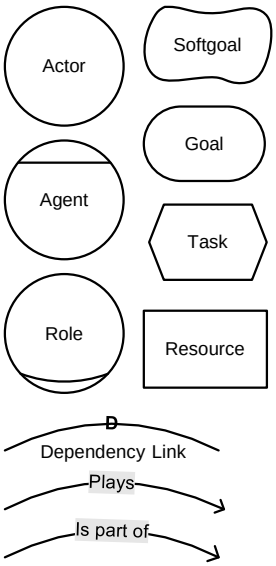

Fig.1. i* Elements and their Graphical Representation

answer these complex communication scenarios and keeping consistency in all the software environments it aggregates, we point to the use of pattern-based agent technology.

Pattern-based MAS development allows indeed mapping (validated) human organizational behavior that is generally applied when facing similar situations in real life to software system behavior. As evoked earlier, in order to structure our MAS architecture, we use a defined organizational style. "*An organization is a consciously coordinated social entity, with a relatively identifiable boundary, that functions on a relatively continuous basis to achieve a common goal or a set of goals.*" [35]. Organization theory is the (social sciences) discipline that studies both structure and design in such social entities. Structure deals with the descriptive aspects while design refers to the prescriptive aspects of a social entity. Organization theory describes how practical organizations are actually structured, offers suggestions on how new ones can be constructed, and how old ones can change to improve effectiveness. To this end, since Adam Smith, schools of organization theory have proposed models and patterns to try to find and formalize recurring organizational structures and behaviors.

Today, organizational structures are primarily studied by two disciplines: *Organization Theory* (e.g., [9,34,44,53]) – that describes the structure and design of an organization and *Strategic Alliances* (e.g., [48,13,21,35,45]), that model the strategic collaborations of independent organizational stakeholders who have agreed to pursue a set of agreed upon business goals. Both disciplines aim to identify and study organizational patterns. These are not just modeling abstractions or structures, rather they can be seen, felt, handled, and operated upon. They have a manifest form and lie in the objective domain of reality as part of the concrete world. A pattern is however not solely a set of execution behaviors. Rather, it exists in various forms at every stage of crystallization (i.e. specification), and at every level of granularity in the organization. The more manifest is its representation, the more the pattern emerges and becomes recognizable – whether at a high or low level of granularity.

[27] propose a set of organizational styles for agent-oriented software engineering; the idea is to map patterns adopted in human organizational theory to the behavior of MAS thus to the internal behavior of software. Among these patterns, we only focus on the joint venture style here. The latter involves an agreement between two or more intra-industry partners to obtain the benefits of larger scale, partial investment and lower maintenance costs. A specific joint management actor coordinates tasks and manages the sharing of resources between partner actors. Each partner can manage and control itself on a local dimension and interact directly with other partners to exchange resources such as data and knowledge. However, the strategic operation and coordination of such an organization, and its actors on a global dimension, are only ensured by the joint management actor in which the original actors possess equity participation. A generic representation of the joint venture style using i* as well as its application in a MOOC context is depicted in [27]. No realization scenario is nevertheless defined nor any design and implementation set in place. The structure is very relevant for building a MOOC platform as an heterogeneous software

environment because it represents the single governance structure over various software environments. This is exactly the same way that a joint venture is set into place for the governance of various companies.

On the basis of the requirements analysis on the one side, and the generic joint venture pattern on the other side, we have built a MAS architecture showing how MOOC organizational roles can be used to support the platform execution. Later on, the architecture needs further design to be able to be implemented through software; this is also supported by a set of patterns aligning the roles of the joint venture with concrete software agents. These patterns are called social patterns [26,25]. In other words, the design involves the description of the system agents behavior in order to fulfill the identified scenarios. The organizational agents from the joint venture applied to the MOOC are relayed/aligned by the software agents in order to execute all of the activities they are required to at run-time (this is depicted in Section 6).

Finally, the agent-design has been implemented using agent technology in the form of a prototype for a proof-of-concept (this is depicted in Section 7).

4 Requirements Definition

This section aims to provide a description of the communication requirements analysis of a MOOC platform. To this end, it depicts standard scenarios. Communication issues are vital for the success of a MOOC with potentially a huge amount of students and a very little support staff. It is impossible for this staff to answer al of the questions or furnish all the material requested. As a consequence, the student community itself has to help in the global effort. Indeed, most of the requests (questions, need for material, ...) can be addressed by other students and very few requests need to be escalated to junior (e.g. Teaching Assistants (TA)) or senior (e.g. professors) support staff. Such scenarios are envisaged and conceptualized in this section.

A Learning Peer (i.e., Learner, Instructor or an Educational Institution) can make a request to another Learning Peer to obtain a Learning Object[1]. Such a request can be related (i) to accessing learning material (e.g. documents, slides, books, videos, ...) not yet provided or not accessible by the requester (e.g., because he does not have access to the system providing it), or (ii) asking for a service concerning learning material (e.g., a question about the course, be evaluated on an exercise, etc.). For the sake of simplicity, we assume here that every time a request for a Learning Object is issued, the addressed Learning Peer can satisfy it in three principal ways: (a) using his/her

[1] Multiple definitions of Learning Objects exist in literature, [6] define a Learning Object as *a representation designed to afford uses in different educational contexts*; this reference is useful for a full typology of the concept. More precisely in the context of our conceptualization, the Learning Object can be considered as a modular resource that can be digitized (so dematerialized) to be used and re-used for the support of learning (and e-learning) activities. Examples are provided further on in the section.

internal knowledge, (b) escalate the request to a known Instructor (another Learning Peer) at an upper-level in the implicit Learning Peer hierarchy that can fulfill it (e.g., TA to Professor), and (c) delegate the answer to one or a set of Learning Peers located at a same or a lower hierarchic level (e.g. an Instructor or set of Learners of the community).

Scenario (a) means that the Learning Peer has himself to furnish the solution or answer the request so that he has to hold/own the requested material or have the ability to furnish the service required to satisfy the request. As a consequence, such an approach minimizes the *time to answer* but requires the knowledge or ability to answer the request; in a word, he has to own the competence to fulfill the request. On the contrary, if (and only if) such competence is not owned by the Learning Peer, he/she can turn to the use of scenario (b). That is, the Learning Peer redirects the request to another known Learning Peers (at an upper rank in their internal organization scale) who can satisfy (or contribute to satisfy) the request. For example, each time a
Learner issues a Learning Object request to Instructor1 (e.g., a TA) that the latter is not able to satisfy due to internal limitations, he distributes the request for a
Learning Object over to Instructor2 (e.g., a more senior TA or a Professor) owning the competencies, material or authority to satisfy the request. In the case none of the lowest level Learning Peers (i.e. students) can fulfill the requests, they must nevertheless be raised by a significant amount of Learning Peers (the threshold can be custom defined) to be addressed to the upper level Learning Peers. The threshold serves as filter for selecting the requests that really need upper-level intervention or validation.

Additionally, an interest of this approach is also that each Instructor can autonomously make his own decisions about how to answer requests at best based on the experience and their pedagogical approach. However, this can only be the case if he owns the authority and leadership to proceed this way; otherwise he/she has to escalate as described previously. Such an approach is convenient for a MOOC composed of members distributed into different geographical areas with lots of Learning Peers involved. The reader should also note that often in such a setting, a central authority (an Educational Institution) would be willing to impose strict (predefined) teaching guidelines that could constitute a drawback to the flexibility in the request satisfaction.

Finally, in scenario (c), an instructor may have the competence to answer the request but not the time to fulfill it. This includes that he could invest his time in a way that would be more rewarding for the Educational Institution (e.g. a Professor that receives a request a TA could satisfy) or just that an Instructor is willing to make other Learning Peers (e.g. Learners) answer the request (as an exercise for them, to open the debate, ...). In a sentence, the Learning Peer delegates the request to a Learning Peer (community) at the same or lower level in the hierarchy. In such cases, the Instructor just makes an ex-post validation of answers to requests.

As an exercise, it is interesting to observe one of the main advantages, in terms of costs, of applying such a collaborative model within a MOOC. While in an independent[2] scenario (a) the request is fulfilled optimally (minimizing the time and at the right cost), scenario (b) increases both the time to answer and the cost to answer (for the Educational Institution) but scenario (c) increases the time to answer but lowers the cost to answer. All in all, for an optimal use of resources, the requests should immediately arrive at the most appropriate level; there is no universal answer to this problem that needs to be evaluated on a case by case basis by intelligent agents. Generally, the adopted tactic is to address the requests at lowest level possible so that only the ones requiring very specific competences are escalated to the (costly) highest levels. This is not optimal in terms of time to answer though and thus a better compromise should be found.

5 Organization-Based Architectural Design

Section 5.1 overviews a MOOC architecture as a joint venture allowing to flexibly deal with an heterogeneous software environment while Section 5.2 depicts each of the internal organizational software agents used by the Software Learning Platform in order to support the software requirements.

5.1 Organizational Style Analysis

Architectural styles are intellectually manageable abstractions of system structures that describe how system components interact and work together. MAS architectures in the context of software engineering can be considered as social organizations composed of autonomous and proactive agents who cooperate with each other to achieve common or private goals [23,30,36]. A key aspect to conduct macro-level architectural design is the specification and use of organizational styles [27]. These are socially-based design alternatives inspired by models and concepts from organizational theories that analyze the structure and design of real-world human organizations (see e.g. [9]). Therefore, taking into account also the requirements analysis results, the proposed MAS architecture, sketched out in Figure 2, has been designed following and adapting the generic joint venture organizational style (depicted in Section 3). The latter is here instantiated to design the specific MOOC architecture in Figure 2. In the rest of this section, it is described through a semi-formal specification.

As already evoked, the proposed system does not aim to substitute the internal educational institution behavior and features. On the contrary it proposes and supports a *learning agent computing approach* to model the collaborative interactions among stakeholders. Indeed, the proposed joint venture style offers the most suitable

[2] An independent scenario (a) means that the scenario is successfully executed without being executed after scenario (b) or (c) or multiple instances of those.

decentralized organization to better fit a platform serving as an abstraction layer over other software environments that has to support the requirements depicted in Section 4. The principal partners of the joint venture are autonomous actors (or agents and roles) who directly exchange services, data, and knowledge with each other similarly to the partners composing a network of decentralized learning agents in a MOOC context (e.g., see [28]). Figure 2 shows that there are several (real-life) actors i.e., Instructor, Educational Institution (e.g., university and high school) and Learner who can play the online avatar role of Learning Peer that constitutes namely the Joint Manager Public Interface interacting with the external (real-life) actors. It allows each of them to make the most of the Learning
Agent Platform (i.e., the Joint Manager Private Interface in canonical form) in order to satisfy their own goals. They indeed depend on the latter to get Positive Feedback and to Migrate (from traditional classroom or on-line teaching) to MOOCs. Similarly, the Learning Agent Platform depends on the Learning Peer to get Confidentiality Ensured as well as software quality attributes such as Bandwidth Optimization or Fault Tolerance. Content Manager, MOOC Coordinator and Learning Objects Facilitator are software
agents, parts of the Learning Agent Platform providing the Learning Peer with learning help and assistance. Hence it is able to fulfill each Learning Peer's request for coordinating tasks, and managing learning resources and outcome that have to be shared among the learners. More precisely, they are in charge of the following tasks for the Learning Agent Platform: the Content ManagerQueries Content, the MOOC CoordinatorOperationalizes MOOC Scheduling and the Learning Object FacilitatorProvides Learning Objects.
These three internal software agents interact with each other and are also responsible for providing resources among each other. The Content Manager provides
Formatted Contents to the MOOC Coordinator. The MOOC Coordinator provides Course Organization to the Content Manager as well as Educational Services to the Learning Object Facilitator. Finally, the Learning Object Facilitator provides both the Content Manager and the MOOC Coordinator with Learning Objects resources.

5.2 System Agents and Organizational Roles

After having overviewed the global architecture as a joint venture, we will now turn to the description of the (intentional) software agents.

System agents (i.e. the Content Manager, MOOC Coordinator and Learning Objects Facilitator in Figure 2) indeed implement the outcomes from

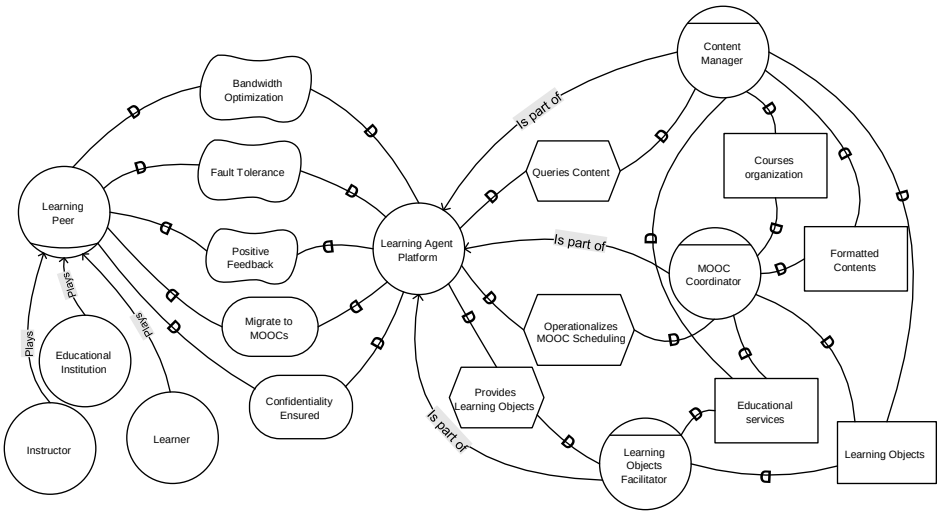

Fig.2. A Massive Open Online Course in a Joint Venture Style

both requirements and organizational-styles analysis, in order to satisfy the (business) organizational roles (Learning Peer) played by human actors (i.e. the Instructor, Educational Institution and Learner) and requiring the Learning Agent Platform system actor. Namely, by means of such agents, all the system requirements for the Learning Agent Platform are satisfied. Hence, they are what we actually intend to design and realize. Such agents will be provided by the software system and supported by means of agent abilities.

At this modeling level, even if the business organizational roles played by the organizational actors (e.g., Instructor) are in general different from the agents of the system-to-be architecture, they easily drive the designer to relate/bridge human organizational needs to agent-based architectural abstractions. These abstractions will be supported by (social) agent patterns as shown in Section 6.

As indicated in Figure 2, a Learning Agent Platform deploys the following agents (due to the joint venture partners capabilities) to support the needs of its human/organizational peer:

The Learning Objects Facilitator (searching and registration). As emerges from Section 4, in scenarios (b) and (c), the involved Learning Peer needs to look for other Learning Peers capable of satisfying a given request. The Learning Obj-
ects Facilitator role provides a learning agent platform with searching and registration capabilities, which allows the platform to get to know other Learning Peer agents with useful skills (to establish new acquaintances able to address Learning Objects requests). Indeed, looking at Figure 2, each agent inside the Learning Agent Platform

depends on the Learning Object Facilitator to perform the task Provide Learning Objects. Moreover, each agent a Learning Agent Platform aggregates (i.e., Content Manager and MOOC Coordinator) depends on the Learning Objects Facilitator through the resource Learning Objects that needs to be furnished. For example, in the prototype presented in this paper, this ability is based on the Facebook group users repository. Such a repository also provides information about the state (e.g., active, disconnected, etc.) of other
Learning Peer agents. The logical grouping of Learning Peer agents (e.g., agents able of the same services/skills), sustained by the repository, forms a Learning Peer domain. Moreover, as depicted in Figure 2, each time a request cannot be internally satisfied, the MOOC Coordinator or the Content Manager could interact with the Learning Object Facilitator agent to get new acquaintances. Specifically, such a behavior is required to cope with environmental constraints introduced by scenario (b). Notice that, in the case of a new Learning Peer request, the Learning Objects Facilitator can also autonomously propagate the request over the peer network without overloading the MOOC Coordinator, e.g., interacting with other platform facilitators. Such a behavior can be required to cope with constraints introduced by scenario (c).

The Content Manager (reformulation and integration). According to the requirements analysis, in terms of environmental constraints, when a given Learning Peer (Instructor) operates in scenarios (b) and (c), he needs to interact with different software systems or platforms used by various Learning Peers. Requests could indeed be issued from a CMS, a LMS, social networks, etc. In other words, we are in the presence of a distributed and heterogeneous information system. In particular, the Learning Agent Platform relies on this agent to perform and coordinate queries targeted to information sources of the same peer or different peers, e.g., as indicated in Figure 2 by means of task dependency Query heterogeneous content and resource dependency Formatted Contents. Therefore, there exists a wellknown data integration problem in distributed, heterogeneous, and dynamic systems. As a consequence, to cope with integration issues, the agent platform can adopt a mediator architecture based on mediator and wrapper agents to access the information sources. For example, it can use one of the several algorithms for answering queries using views (e.g., see [37]). Therefore, the Content Manager provides a Learning Agent Platform with reformulation and integration capabilities. Using these, an agent platform can reformulate the initial request in terms of data management operations targeted at selected sources and respecting database constraints.

The MOOC Coordinator (strategy generation). The MOOC Coordinator agent is required to correctly coordinate collaboration activities among decentralized Learning Peers, i.e., virtual-learning members. For instance, when a failure results from the Learning Agent Platform inability to satisfy a request locally, the MOOC Coordinator can help to build a cooperation strategy in order to overcome the underlying failure. In particular, the system prototype MOOC Coordinator currently deals with the principal failures that affect virtual learning scenarios, such as: *inability to answer a request for a*

Learning Object, *instructors unavailability* and *pedagogical approach implementation*. Specifically, the MOOC Coordinator manages plans (workflows) composed of actions in order to operationalize the MOOC schelduling, i.e. *ensure requests fulfillment*, *query information sources*, *edict guidelines to be followed*, etc. To this end, such an agent can rely on the well-known BDI architecture [4,43,5, 41,49]. According to this architecture, the MOOC Coordinator represents the environment status in terms of facts (i.e., the beliefs) and the received requests in terms of goals (i.e., the desires). Moreover, he chooses the more convenient behavior (i.e., the intention), among a set of plans, to achieve the current goal. Finally, each MOOC Coordinator has the responsibility of coordinating the internal activity required to keep all the learning repositories up to date. In the system to-be designed, the MOOC

Coordinator relies on the Content Manager in order to inquire the Learning Peer's internal information sources (required to update its beliefs), e.g., repositories to get the status of specific Learning Objects. To this end, Figure 2 and 4 indicate such relations by means of resource dependencies Formatted Contents and Courses Organization.

6 Detailed Design of the MOOC architectured as a Joint Venture

The joint venture style applied to the MOOC architecture in Section 5 defines the logical behavior of the system-to-be developed. Indeed, every time an organizational style is applied, it allows easily pointing up, to the system designer, the organizational actors and roles that need to be supported by the system. The following step in the MAS development thus requires mapping and detailing the system agents and organizational roles to specific software agents and characterize their behavior. Namely, each role in Figure 2 is much closer to the real organizational actor behaviors than software agent behaviors that we aim to achieve. As a consequence, the organizational characterization that we have made allowed determining the MAS global structure in terms of actors, roles, and their intentional relationships. Now to build the software and make it executable, a deeper analysis/modeling of these entities is required. They will then become software entities behaving at run-time to support requirements. As evoked in Section 3, in order to achieve this next step, other patterns are available. These are called *social patterns* for detailed design. They focus on social and intentional aspects that are recurrent in MAS or cooperative systems. Moreover, similarly to organizational styles, social patterns are generic structures that define how (a small number of) agents interact together in order to fulfill their obligations. Social patterns are applied here to depict the system behavior. Roughly speaking they map the organizational behavior through software entities. The full understanding of agent-oriented design is nevertheless not required to understand how they work. The application of the design patterns is depicted in the rest of this section.

The idea that governs this design phase, consists of two main steps: (a) characterizing each system actor in terms of agents and their interdependencies needed to accomplish the organizational roles; (b) organizing such agents in recurrent structures, named social patterns, in order to allow designers to reuse design experience and knowledge during the whole system development process. According to (a) and (b), we modeled each organizational system agent, i.e., Content Manager, MOOC Coordinator, Learning Objects Facilitator, and Learning Agent Platform, in terms of agents and related agent structures. In particular, this section focuses on the detailed design of the actors Learning Agent Platform and Content Manager.

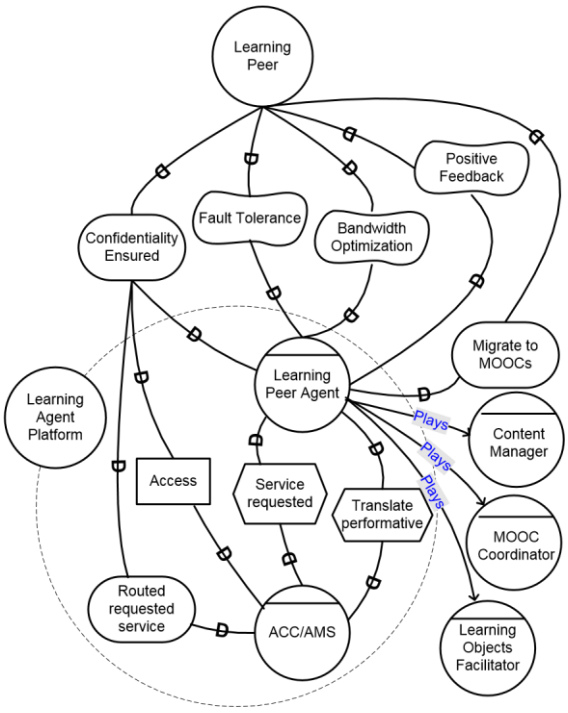

Fig.3. The Learning Agent Platform as an Embassy Agent Pattern

In Figure 3, the embassy agent pattern [26,25] has been applied to build the core part of the agent system: the Learning Agent Platform actor. In this pattern, an embassy agent (ACC/AMS) routes and translates in both directions a service requested by an external role (represented by instances of the Learning Peer) to local agents (the Learning Peer Agent). Specifically, the actor Learning Agent Platform, depicted in Figure 2 by a round shape, coordinates and manages Learning Peer's requests by distributing them to the correct role. Notice that, such roles must be played by the agent Learning Peer Agent (circle with a line on the top) depicted in Figure 3 since it is a software agent

in charge of all the tasks delegated by the corresponding Learning Peer (see [39] for details). Figure 3 explicitly shows the PLAYS link dependencies. A Learning Peer depends on the agent ACC/AMS to submit messages and gain access to the agent platform resources, as follows. Each Learning Peer depends on the *Agent Communication Channel* (*ACC*) to route messages over different protocols (goal dependency Routed requested services). Again, each Learning Peer depends on the *Agent Management System* (*AMS*) to manage agent life-cycle (or agent state) and provide access grant to the Learning Agent Platform (resource dependency Access). Moreover, each time an agent of the Learning Agent Platform (i.e., Learning Peer Agent roles) needs to interact with an external one (i.e. belonging to another platform) that exploits different performatives, a translation ability is required to the router agent (ACC).

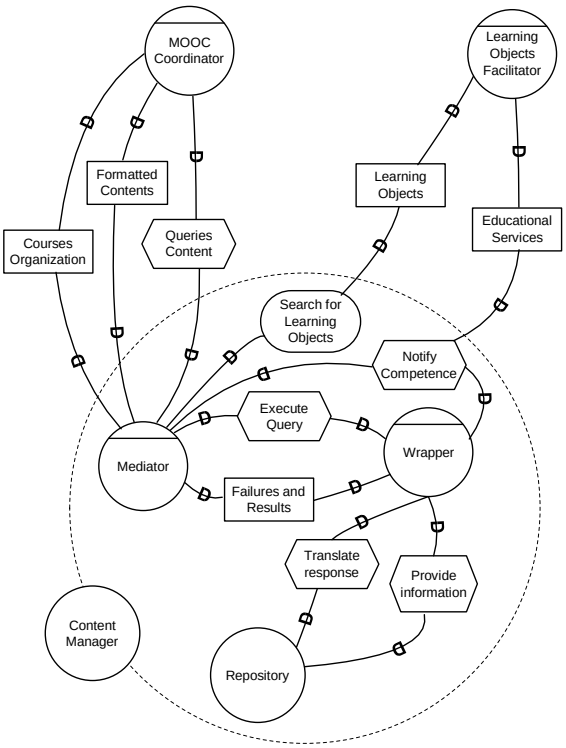

Fig.4. The Content Manager as a Mediator Agent Pattern.

Figure 4 gives another example about how to model organizational roles by means of agent patterns. Specifically, the mediator agent pattern [26,25] is adopted to correctly match the Content Manager requirements arosen from the previous (socialbased) organizational analysis of Section 5. In this pattern, a Mediator agent

coordinates the cooperation of performer agents – Wrappers – to satisfy the request of an initiator agent – the MOOC Coordinator. For the last 20 years, mediators-based systems have become the reference architecture to integrate both structured and semistructured data (e.g., see [51,38]). Into our system, the Mediator plays a key role because he makes an intelligent interface by dealing with representation and abstraction problems that one must face when trying to use data and knowledge resources. Specifically, each time a request arrives to the MOOC Coordinator, he needs to extract all the information about the requested Learning Object (indicated in Figure 4 by the Queries Content and the Formatted Content dependencies) to effectively deal with the choice of the most convenient execution plan. Hence, he relies on the Content Manager. Indeed, the latter must be able both to reformulate the MOOC Coordinator request into other sub-requests, and, vice-versa, to integrate the answers into a single and coherent answer to the MOOC Coordinator (thus Formatted Content). That is, MOOC Coordinator depends on the agent Mediator to furnish the Query results content. Moreover, as better detailed in the following, each Mediator agent can rely on one or more wrapper agents to fulfill the Query content task dependency since, in general, content providers (actor Repository) are not agent-based, they are LMS, CMS, social networks, etc. they dialogue with.

As introduced in Section 5.2, the Learning Object Facilitator supports a list of service descriptions where mediators and wrappers can register their Educational Services (performing task dependency Notify Competence[3]) and where each Mediator can search for new Learning Objects. Finally, notice that the agents that appear in Figure 4 can belong to different Learning Agent Platforms. Indeed, according to Figure 2 and 3, each Content Manager can interact with different Learning Peers, specifically peers who play the role of Content Manager as well. As a consequence, a Mediator of a Learning Agent Platform can interact with Wrappers, Learning Objects Facilitators, Mediators, and MOOC Coordinators of different platforms.

7 Validation

This section presents the prototype developed as a proof-of-concept of the MOOC platform developed in this paper and focuses on the experience that the user can have on its basis.

Concretely, the prototype is implemented around a two-tiered architecture: (i) the Graphical User Interface (GUI) that is represented through a web-based application constitutes the client tier and (ii) the MAS that encapsulates the core system logic constitutes the server tier. In order to avoid getting too deeply in the technical details, the technologies and programming languages used for the implementation are

[3] The *Competence* represents what it is able to do to furnish (or contribute to furnish) an answer to a defined Learning Object request; it is structured through a defined typology.

depicted in appendix A, the dialogue between the GUI and the MAS are overviewed in appendix B.

7.1 Integration with Third Party Platforms: The Case of Facebook

Figure 5 presents the interfacing possibilities of a native MOOC platform with the GUI of Facebook; the contents are synchronized at the level of the MAS platform.

When a MOOC is created through our MAS platform, a specific Facebook page is created for that course and contents generated for the course in the traditional MOOC VLE are pushed in the corresponding Facebook page. When a Facebook Learning Peer registers to that course, he also automatically joins the Facebook page and gets notified of updated content through the Facebook notification system. Learning Peers can interact with other Learning Peers using the Facebook course page via the native posting and commenting systems. They can also use the *#hashtag* function with the name of the corresponding MOOC. This way on the *#hashtag* feed page they get an overview of latest actions related to the MOOC. More importantly, all communications are centralized in a Facebook group for the MOOC (also generated when the MOOC is created in Facebook); communication is detailed in the next section. All posts and comments on the Facebook page are systematically synchronized with the main Facebook course page and with the native MOOC platform.

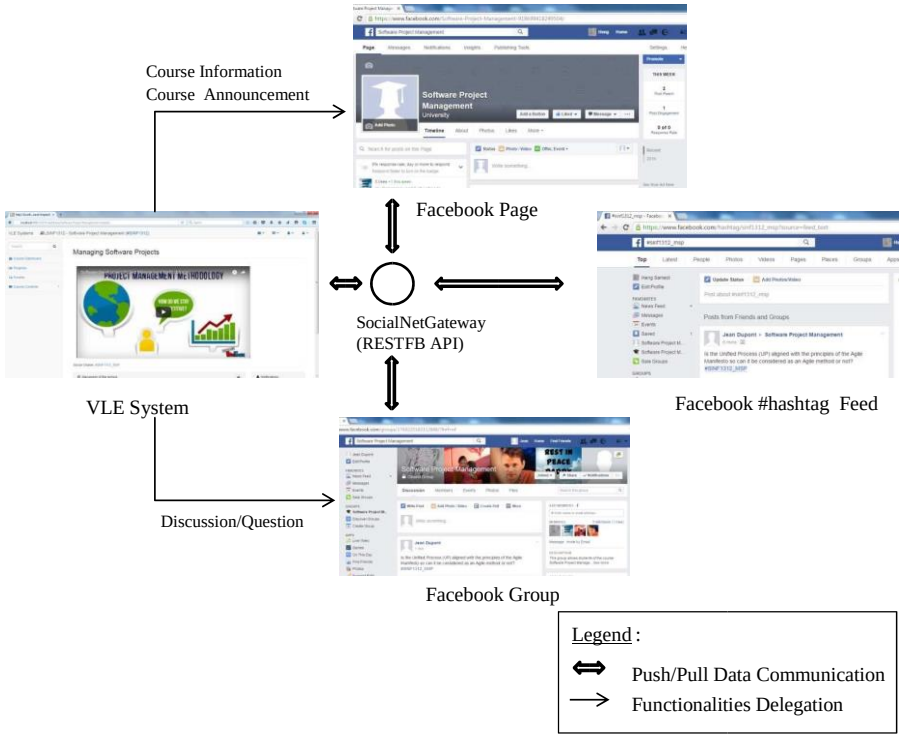

Fig.5. Mechanisms used into Facebook to (re)present the MOOC.

7.2 An Interface for Supporting the Requirements Scenarios

Let us take the example of a Learning Peer using Facebook for interaction with our MOOC.

Figures 6, 7 and 8 show the different views of the requests for Learning Objects and the interaction of Learning Peers. Figures 6 and 7 are in the native MOOC environment while Figure 8 represents the counterpart of the communication scenario in Facebook. In this perspective, within these Figures, we distinguish a few boxes – (1), (2), (3) and (4) – that correspond to the same elements but in different views. Concretely, box (1) existing in all the figures shows a question raised by a student through the Facebook Group (Figure 8). When this action is performed, it is automatically pulled in the native MOOC environment (Figures 6 and 7). Box (2) existing in Figure 6 shows that an Instructor Learning Peer can delegate or escalate a request to other Learning Peers who can possibly fulfill the request. Box (3) existing in Figures 7 and 8 allows us to visualize the answer of a Learning Peer (whatever its level)

to the request of another Learning Peer (e.g. a Student answers the question of another Student). Finally, within box (4) existing in Figure 7 and 8, the Instructor can validate an answer; this is done in Facebook by an instructor using the *@function*.

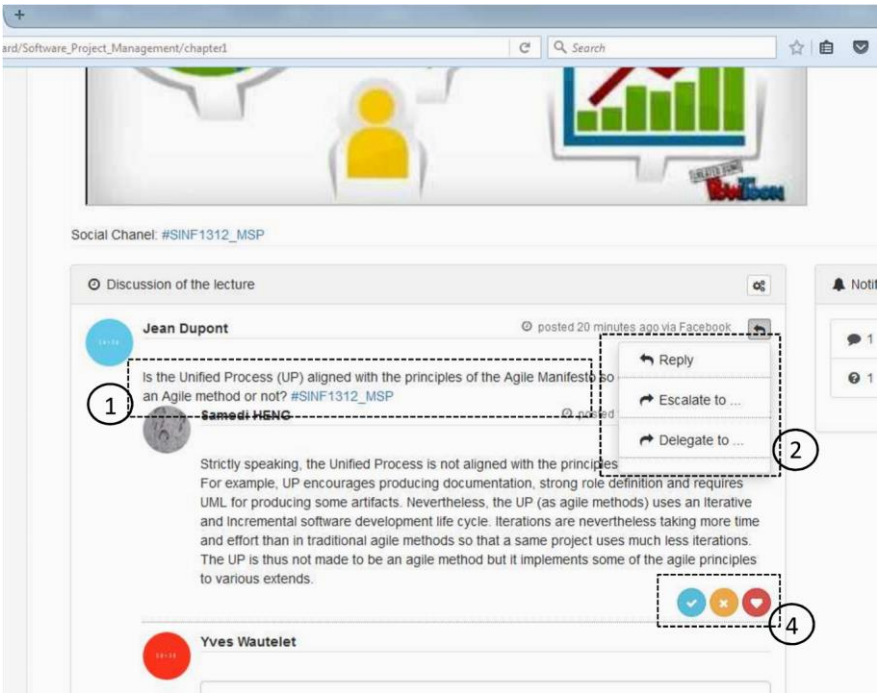

Fig.6. Instructor View of a Request for Learning Objects in the Native VLE.

8 Conclusion

The evolution of e-learning activities has led to the creation of VLE with more and more functions offering real perspectives for effective distance learning. As an ultimate

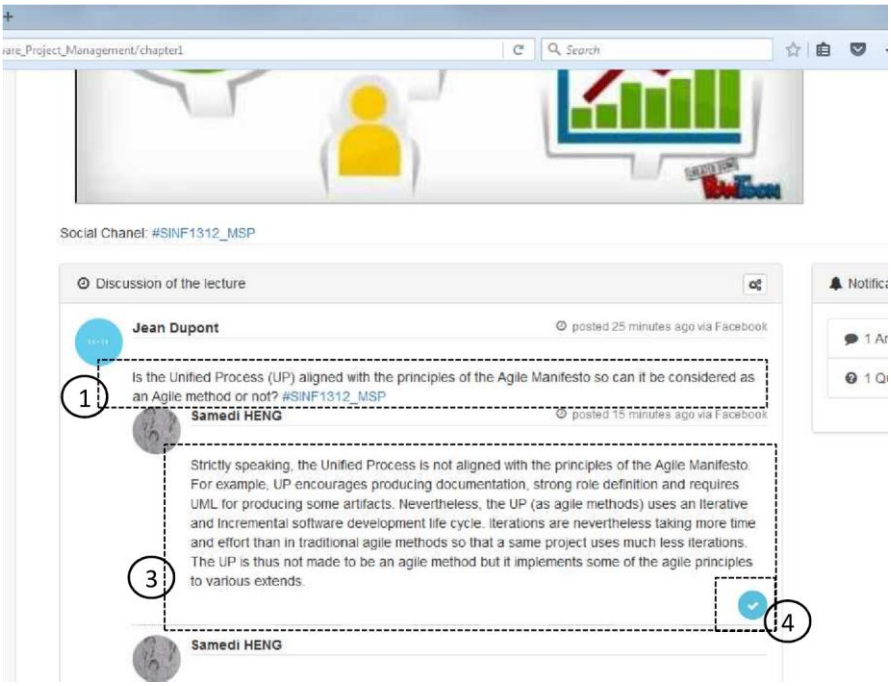

Fig.7. Student View of a Request for Learning Objects in the Native VLE.

development we find MOOCs that can encompass thousands of learners needing to be actively supported by instructors. Indeed, without proper support, a MOOC is only a collection of learning material where interaction is limited to pre-defined scenarios or “Frequently Asked Questions” to support users. Literature relates that in average only 10 to 20% of the learners keep their motivation to follow the MOOC until the end and students performing the best are the ones generating the most communications. In this context requests for Learning Objects need to be carefully managed and easily accessed if we want to raise motivation of learners and stimulate them to communicate.

Within this paper, we have identified a few scenarios where requests for Learning Objects can be issued from learners and addressed in an heterogeneous software environment to instructors. We thus suggest building a MOOC that works as an upper layer to classical VLE bringing heterogeneous learning communities together without requiring them to log in a third-party platform. In order to deal with the complexity of such an environment at run-time, pattern-based agent technology has been used. Applying the joint venture style allows us to deal with the complexity of

communications in heterogeneous software environments in the fashion that various companies set-up a joint governance. Software agents – playing the roles found in the joint venture and instanciated to the MOOC context – then behave/organize as real life entities.

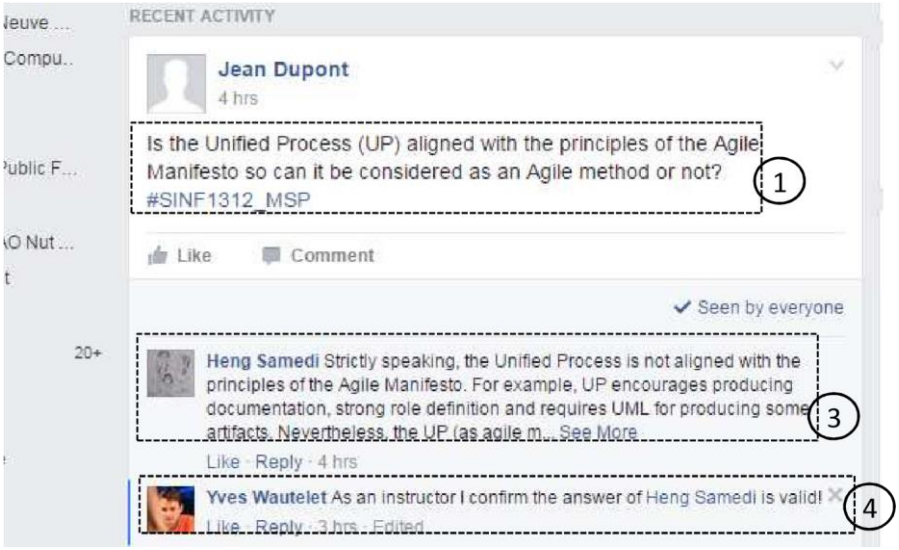

Fig.8. Learning Peer View of a Request for Learning Objects in Facebook.

The paper showed the feasibility of the proposal by developing a usable prototype. Its interaction with Facebook allows synchronizing requests for learning material automatically with the prototype. This allows Facebook users to actively follow the MOOC, be notified of every updates immediately from that third party software environment, answer questions, validate answers, overview latest requests, etc. The more active the learning community, the more communication is generated through a snowball effect.

Contributions of the software platform developed in this paper are located in multiple research domains: on the one hand to the e-learning field – by showing how traditional VLE can be outdated by their aggregation into one and single portal working in the cloud – and on the other hand to MAS development theory – by showing the applicability of agent-oriented development and its related patterns in the field of e-learning.

Future work includes the deployment of the platform on a large scale. For this purpose, a MOOC is currently under development in the research department. When the MOOC is deployed, we will perform an in-depth evaluation of the perceived user experience by students and instructors.

A Technologies and Programming Languages Used

For the development of the GUI, the *AngularJS* framework[4] [10] has been used in combination with *Bootstrap* [5] [2]. AngularJS is a *Model-View-Controller* (*MVC*) framework based on the *JavaScript* language [14] that facilitates the development and the maintenance of single page (see [33,32]) and rich (see [16] internet applications. AngularJS uses extra tags embedded within HTML pages [20] to show the value of JavaScript variable. The tag is a variable used in JavaScript; it represents the model in the MVC framework and, in addition, the model could also be a *JavaScript Object Notation* (*JSON*) object [8]. The binding mechanism implies that when a value of a JavaScript variable is changed, the HTML page is automatically updated with the new value. Bootstrap is a *Cascading Style Sheets* (*CSS*) framework [20] based on a grid system [6] initially developed by Twitter; it allows building an ergonomic end-user interface compatible with multiple platforms. The joint use of these frameworks brings significant advantages because Bootstrap enhances AngularJS user interface but does not provide the JavaScript library allowing building rich and single page applications (as AngularJS does).

For the server side, the MAS has been implemented using the *Java Agent DEvelopment Framework* (*JADE*) [3]. JADE is a framework used to implement MAS which conforms to the FIPA standard (see [42]). JADE simplifies the MAS development while ensuring standard compliance through a comprehensive set of system functions and their related agents. JADE also uses the Java language to implement agents and uses ACLMessages (see [3]) to communicate between agents. Jade can nevertheless not directly communicate with a web-browser by the use of the HTTP protocol but provides a gateway to communicate with any program written in java with the help of a Java Object. It is therefore necessary to use a Java web server as a bridge to allow web clients to communicate to agents: in our case, we have used the Tomcat web server with Java
Servlet.

[4] https://angularjs.org

[5] http://getbootstrap.com

[6] A grid system allows the content of a page to systematically adjust itself to fit the size of the screen vertically and horizontally with consistent information for proper navigation [2].

To communication with Facebook through the Java language, we use *RestFB API*[7], a free Facebook Graph API[8] client also written in Java language. The SocialNetGateway indeed uses the RestFB API to push and pull information between the MOOC native platform and Facebook. When there is a change in the MOOC native platform, the SocialNetGateway receives the request from the Learning Agent Platform and pushes the update to Facebook. The reverse process does not work unfortunately: Facebook does not provide any push mechanism for an external platform. Therefore, the SocialNetGateway sets a timer to pull information from Facebook and push the update to the native MOOC platform. As a consequence, an update in Facebook is not instantly synchronized with native MOOC platform; the update time depends on the value of the timer set in SocialNetGateway.

B Dialogue between the GUI and the MAS

For an effective execution of the software we need to allow the web-client to communicate directly with the agents present in the Jade platform. When an action is performed at the level of the GUI, the agent concerned with the transaction (see Section 6) receives what the web-client has sent. We propose in this implementation an encapsulation mechanism that allows the web-client to send or receive content to/from the concerned agent directly. This means that the agents send and receive the requests composed within the client (meaning at the level of the GUI) in real-time. We use the JSON format for communication between the GUI web-client and the MAS since this format is popular in web technologies and light weight when compared to other formats like XML. In addition, it allows the GUI client to directly update its interface after getting the data from the corresponding agent. In order to achieve this, the request of the client is, as a first step, encapsulated in the HTTP request. When the Servlet receives any HTTP request from the client, it reads the content of this request in JSON data format and writes it into a Java Object. Then, as a second step, it sends that object to the JADE gateway. Finally, the Gateway reads the JSON data from the Java Object sent and builds the ACLMessage using the JSON data. The ACLMessage is sent to the Learning Agent Platform. The Learning Agent Platform can route the request to the corresponding agent or treat the message locally depending on the data present in the request (see Section 6).

For simplicity and efficiency, we propose the structure for the client requests detailed in Figure 9. Every request is indeed structured around four dimensions: *destination*, *source*, *operation*, and *content*. The request itself is also structured on the basis of the JSON format. Each time, the Learning Agent Platform receives an

[7] http://restfb.com/

[8] https://developers.facebook.com/docs/graph-api

ACLMessage, it de-encapsulates the content and reads the information of the destination. If the value of the destination is *null*, it means that message has to be treated by Learning Agent Platform. This request could be for example a login request. If not, the Learning Agent Platform has to forward that request to the corresponding agent following the value of the destination by using the ACLMessage with the request inside the message content. In order to facilitate the routing task, the values refereed to for the destination and source are, in Jade, the *Agent ID* (*AID*). Each time a new agent is created, i.e. a Learning Peer Agent, it communicates its AID to the web-client. By doing this, the agent receives the messages composed by the client and structured as exposed in Figure 9.

Destination	Source	Operation	Content

```
Request = '{d : '', s : '',
                    op: '',
                   content:[{}]
                   }';
```

Fig.9. Client request in JSON format.

As mentioned earlier, the goal of the MAS architecture is to allow aggregating content and Learning Peer communities from heterogeneous software platforms like social networks to enroll to the MOOC and request or consult Learning Objects. In order to achieve this, the Learning Agent Platform additionally provides two other kinds of gateways dedicated to the communication with external software platforms. SocialNetGateway is a JADE agent in charge of exchanging course content with social networks like Facebook. Whereas MOOCGateway is in charge of exchanging course content with other VLE. Figure 10 exposes the architecture of our MAS platform communicating with other software environments.

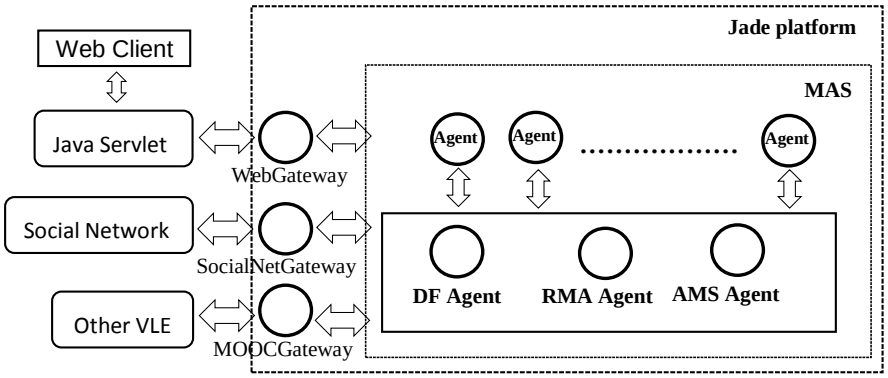

Fig.10. MOOC Deployment Architecture for Integration of Heterogeneous Platforms.

References

1. Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R.H., Konwinski, A., Lee, G., Patterson, D.A., Rabkin, A., Stoica, I., Zaharia, M.: A view of cloud computing. Commun. ACM 53(4), 50–58 (2010), http://doi.acm.org/10.1145/1721654.1721672
2. Balasubramanee, V., Wimalasena, C., Singh, R., Pierce, M.E.: Twitter bootstrap and angularjs: Frontend frameworks to expedite science gateway development. In: 2013 IEEE International Conference on Cluster Computing, CLUSTER 2013, Indianapolis, IN, USA, September 23-27, 2013. p. 1. IEEE Computer Society (2013), http://dx.doi.org/ 10.1109/CLUSTER.2013.6702640
3. Bellifemine, F., Caire, G., Greenwood, D.: Developing multi-agent systems with JADE, vol. 5. Wiley (2007)
4. Bratman, M.E.: Intention, Plans, and Practical Reason. Cambridge University Press (1999)
5. Casali, A., Godo, L., Sierra, C.: A graded BDI agent model to represent and reason about preferences. Artif. Intell. 175(7-8), 1468–1478 (2011)
6. Churchill, D.: Towards a useful classification of learning objects. Educational Technology Research and Development 55(5), 479–497 (2007)
7. Coerderoi, R., Vas, A.: Mooc fondements de la strategie d'entreprise (2015),´ https:// www.facebook.com/moocstrat/timeline
8. Crockford, D.: The application/json Media Type for JavaScript Object Notation (JSON). IETF RFC 4627 (2006), https://rfc-editor.org/rfc/rfc4627.txt
9. Daft, R.: Organization Theory and Design. Cengage Learning (2012)
10. Darwin, P.B., Kozlowski, P.: AngularJS web application development. Packt Publishing (2013)
11. Dillenbourg, P., Fox, A., Kirchner, C., Mitchell, J.C., Wirsing, M.: Massive open online courses: Current state and perspectives (dagstuhl perspectives workshop 14112). Dagstuhl Manifestos 4(1), 1–27 (2014), http://dx.doi.org/10.4230/DagMan.4.1.1

12. Dillenbourg, P., Schneider, D., Synteta, P.: Virtual learning environments. In: 3rd Hellenic Conference" Information & Communication Technologies in Education". pp. 3–18. Kastaniotis Editions, Greece (2002)
13. Dussauge, P., Garrette, B.: Cooperative Strategy: Competing Successfully Through Strategic Alliances. Wiley and Sons (1999)
14. Flanagan, D.: JavaScript: The Definitive Guide Activate Your Web Pages. O'Reilly Media, Inc., 6th edn. (2011)
15. Fong, W.W.: From web 2.0 to classroom 3.0. In: Kwan, R., Fong, J., Kwok, L.F., Lam, J. (eds.) Hybrid Learning - 4th International Conference, ICHL 2011, Hong Kong, China, August 10-12, 2011. Proceedings. Lecture Notes in Computer Science, vol. 6837, pp. 298– 305. Springer (2011)
16. Fraternali, P., Rossi, G., Sanchez-Figueroa, F.: Rich internet applications. IEEE Internet´ Computing 14(3), 9–12 (2010), http://doi.ieeecomputersociety.org/10. 1109/MIC.2010.76
17. Gillani, N., Eynon, R.: Communication patterns in massively open online courses. The Internet and Higher Education 23, 18–26 (2014)
18. Gillani, S.A., Ko, A.: Incremental ontology population and enrichment through semanticbased text mining: An application for IT audit domain. Int. J. Semantic Web Inf. Syst. 11(3), 44–66 (2015), http://dx.doi.org/10.4018/IJSWIS.2015070103
19. Gluz, J., Vicari, R.M., Passerino, L.: The socio-cultural approach to software engineering: an application on the analysis and design of a semantic platform for learning objects. In: First International Workshop on Social Computing in Digital Education (SOCIALEDU 2015), Stanford, CA, USA, Revised Selected Papers. Communications in Computer and Information Science, vol. 606. Springer (2015)
20. Goldstein, A., Lazaris, L., Weyl, E.: HTML5 & CSS3 for the Real World. sitepoint (2015)
21. Gomes-Casseres, B.: The alliance revolution: the new shape of business rivalry. Harvard University Press (1996)
22. Hew, K.F., Cheung, W.S.: Students and instructors use of massive open online courses (moocs): Motivations and challenges. Educational Research Review 12, 45–58 (2014)
23. Hu, C., Mao, X., Li, M., Zhu, Z.: Organization-based agent-oriented programming: model, mechanisms, and language. Frontiers of Computer Science 8(1), 33–51 (2014), http:// dx.doi.org/10.1007/s11704-013-2345-6
24. Ivanovic, M., Jain, L.C. (eds.): E-Learning Paradigms and Applications - Agent-based Approach, Studies in Computational Intelligence, vol. 528. Springer (2014)
25. Kolp, M., Wautelet, Y., Faulkner, S.: Social-centric design of multi-agent architectures. In: Yu, E., Giorgini, P., Maiden, N., Mylopoulos, J. (eds.) Social Modeling for Requirements Engineering. MIT Press (2011)
26. Kolp, M., Faulkner, S., Wautelet, Y.: Social structure based design patterns for agentoriented software engineering. IJIIT 4(2), 1–23 (2008), http://dx.doi.org/10. 4018/jiit.2008040101
27. Kolp, M., Wautelet, Y.: Human organizational patterns applied to collaborative learning software systems. Computers in Human Behavior 51, 742–751 (2015), http://dx.doi. org/10.1016/j.chb.2014.11.094

28. Kop, R.: The challenges to connectivist learning on open online networks: Learning experiences during a massive open online course. The International Review of Research in Open and Distance Learning 12(3), 19–38 (2011)
29. Lytras, M.D., Mathkour, H.I., Abdalla, H.I., Al-Halabi, W., Ya´nez-M˜ arquez, C., Siqueira,´ S.W.M.: An emerging - social and emerging computing enabled philosophical paradigm for collaborative learning systems: Toward high effective next generation learning systems for the knowledge society. Computers in Human Behavior 51, 557–561 (2015), http://dx. doi.org/10.1016/j.chb.2015.06.004
30. Mao, X., Yu, E.S.K.: Organizational and social concepts in agent oriented software engineering. In: Odell, J., Giorgini, P., Muller, J.P. (eds.) Agent-Oriented Software Engineering V, 5th International Workshop, AOSE 2004, New York, NY, USA, July 19, 2004, Revised Selected Papers. Lecture Notes in Computer Science, vol. 3382, pp. 1–15. Springer (2004)
31. Masud, M.A.H., Huang, X.: An e-learning system architecture based on cloud computing. system 10(11) (2012)
32. Mesbah, A., van Deursen, A.: Migrating multi-page web applications to single-page ajax interfaces. In: Proceedings of the 11th European Conference on Software Maintenance and Reengineering. pp. 181–190. CSMR '07, IEEE Computer Society, Washington, DC, USA (2007), http://dx.doi.org/10.1109/CSMR.2007.33
33. Mikowski, M., Powell, J.: Single Page Web Applications: JavaScript End-to-end. Manning Publications Co., Greenwich, CT, USA, 1st edn. (2013)
34. Mintzberg, H.: Structure in fives : designing effective organizations. Prentice-Hall (1992)
35. Morabito, J., Sack, I., Bhate, A.: Organization modeling : innovative architectures for the 21st century. Prentice Hall (1999)
36. Mylopoulos, J., Kolp, M., Giorgini, P.: Agent-oriented software development. In: Vlahavas, I.P., Spyropoulos, C.D. (eds.) Methods and Applications of Artificial Intelligence, Second Hellenic Conference on AI, SETN 2002. Thessaloniki, Greece, April 11-12, 2002, Proceedings. Lecture Notes in Computer Science, vol. 2308, pp. 3–17. Springer (2002)
37. Panti, M., Spalazzi, L., Penserini, L.: Cooperation strategies for information integration. In: Batini, C., Giunchiglia, F., Giorgini, P., Mecella, M. (eds.) Cooperative Information Systems, 9th International Conference, CoopIS 2001, Trento, Italy, September 5-7, 2001, Proceedings. Lecture Notes in Computer Science, vol. 2172, pp. 123–134. Springer (2001)
38. Papakonstantinou, Y., Garcia-Molina, H., Ullman, J.D.: Medmaker: A mediation system based on declarative specifications. In: Su, S.Y.W. (ed.) Proceedings of the Twelfth International Conference on Data Engineering, February 26 - March 1, 1996, New Orleans, Louisiana. pp. 132–141. IEEE Computer Society (1996), http://dx.doi.org/10. 1109/ICDE.1996.492097
39. Penserini, L., Liu, L., Mylopoulos, J., Panti, M., Spalazzi, L.: Cooperation strategies for agent-based P2P systems. Web Intelligence and Agent Systems 1(1), 3–21 (2003), http://content.iospress.com/articles/ web-intelligence-and-agent-systems-an-international-journal/ wia00002
40. Pireva, K., Kefalas, P., Dranidis, D., Hatziapostolou, T., Cowling, A.: Cloud e-learning: A new challenge for multi-agent systems. In: Agent and Multi-Agent Systems: Technologies and Applications, pp. 277–287. Springer (2014)
41. Pokahr, A., Braubach, L., Haubeck, C., Ladiges, J.: Programming BDI agents with pure java. In: Muller, J.P., Weyrich, M., Bazzan, A.L.C. (eds.) Multiagent System Technologies - 12th¨

German Conference, MATES 2014, Stuttgart, Germany, September 23-25, 2014. Proceedings. Lecture Notes in Computer Science, vol. 8732, pp. 216–233. Springer (2014)
42. Poslad, S.: Specifying protocols for multi-agent systems interaction. TAAS 2(4) (2007), http://doi.acm.org/10.1145/1293731.1293735
43. Rao, A.S., Georgeff, M.P.: BDI agents: From theory to practice. In: Lesser, V.R., Gasser, L. (eds.) Proceedings of the First International Conference on Multiagent Systems, June 12-14,
1995, San Francisco, California, USA. pp. 312–319. The MIT Press (1995)
44. Scott, W.: Organizations: rational, natural, and open systems. Prentice Hall (1998)
45. Segil, L.: Intelligent business alliances : how to profit using today's most important strategic tool. Times Business (1996)
46. Tibaut, A., Rebolj, D., Menzel, K., Jardim-Gonc¸alves, R.: Inter-university virtual learning environment. In: Ivanovic and Jain [24], pp. 97–119
47. Ting, I.H., Wu, W.J., Kao, H.T., Wang, D.: An implementation of online learning and course management system based on facebook. In: Learning Technology for Education in Cloud, Communications in Computer and Information Science, vol. 533, pp. 208–218. Springer (2015)
48. Todeva, E., Knoke, D.: Strategic alliances and models of collaboration. Management Decision 43(1), 123–148 (2005)
49. Wautelet, Y., Kolp, M.: Business and model-driven development of bdi multi-agent systems. Neurocomputing 182, 304321 (2016), http://dx.doi.org/10.1016/j.neucom.
2015.12.022
50. Weller, M.: Virtual learning environments: Using, choosing and developing your VLE. Routledge (2007)
51. Wiederhold, G.: Mediators in the architecture of future information systems. IEEE Computer
25(3), 38–49 (1992), http://dx.doi.org/10.1109/2.121508
52. Wulf, J., Blohm, I., Leimeister, J.M., Brenner, W.: Massive open online courses. Business & Information Systems Engineering 6(2), 111–114 (2014), http://dx.doi.org/10.
1007/s12599-014-0313-9
53. Yoshino, M., Rangan, U.S.: Strategic alliances : an entrepreneurial approach to globalization. Harvard Business School Press (1995)
54. Yu, E., Giorgini, P., Maiden, N., Mylopoulos, J.: Social Modeling for Requirements Engineering. MIT Press (2011)