Behaviour & Information Technology

ISSN: 0144-929X (Print) 1362-3001 (Online) Journal homepage: http://www.tandfonline.com/loi/tbit20

Designing an MOOC as an agent-platform aggregating heterogeneous virtual learning environments

Yves Wautelet, Samedi Heng, Manuel Kolp, Loris Penserini & Stephan Poelmans

To cite this article: Yves Wautelet, Samedi Heng, Manuel Kolp, Loris Penserini & Stephan Poelmans (2016) Designing an MOOC as an agent-platform aggregating heterogeneous virtual learning environments, Behaviour & Information Technology, 35:11, 980-997, DOI: 10.1080/0144929X.2016.1212095

To link to this article: http://dx.doi.org/10.1080/0144929X.2016.1212095

Published online: 09 Sep 2016.

BEHAVIOUR & INFORMATION TECHNOLOGY, 2016
VOL. 35, NO. 11, 980–997
http://dx.doi.org/10.1080/0144929X.2016.1212095

Taylor & Francis
Taylor & Francis Group

Designing an MOOC as an agent-platform aggregating heterogeneous virtual learning environments

Yves Wautelet[a], Samedi Heng[b], Manuel Kolp[b], Loris Penserini[c] and Stephan Poelmans[a]

[a]Faculty of Economics and Business, KU Leuven, Brussels, Belgium; [b]Louvain School of Management, Université catholique de Louvain, Louvain-la-Neuve, Belgium; [c]Istituto Tecnico Commerciale 'Cesare Battisti', Bolzano, Italy

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 drastically 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. These nevertheless remain different systems but, by using these heterogeneous software environments together, the possibilities for interaction could be multiplied. That is why, this paper suggests to build 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 such as social networks to furnish an easy access to the MOOC of their large user communities. The MAS design has been architectured around a real-life organisational pattern – the joint venture – allowing one to deal with the complexity of heterogeneous software environments in a manner that real-life companies set up joint governance. Communication scenarios issued of 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.

ARTICLE HISTORY
Received 24 February 2016
Accepted 8 July 2016

KEYWORDS
Multi-agent systems; organisational 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 Dillenbourg, Schneider, and Synteta 2002; Weller 2007), 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 Fong 2011) that should have no physical boundaries and will be accessible anytime and anywhere.

A lot of new platforms such as *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 among each other and potentially interested in learning activities. An integration of these platforms with one or several VLE could thus enable to dispose of huge user communities without the 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

CONTACT Yves Wautelet yves.wautelet@kuleuven.be

environment designed as an upper layer to classical VLE such as*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 such as 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 up a *Massive Open Online Course* (*MOOC*).

An MOOC is an online course aimed at unlimited participation and open access via the web (see Dillenbourg et al. 2014; Wulf et al. 2014). 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 such as 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 wastes of time for instructors (and thus wastes 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 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 to 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 social-based structures (Lytras et al. 2015). 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 leads to its higher 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 organisational 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 behaviour on the behaviour of individuals organising 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 an MOOC (see Section 2);
- to architecture the realisation of these scenarios around a collaborative MAS-platform (following organisational 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' behaviour in order to synchronise communication content over multiple software environments such as social networks, thus acting as an upper layer over classical VLE. An MOOC that is accessible in a software environment the user is already familiar with positively impacts 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 synchronised with third-party platforms such as Facebook with the use of an 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 behaviour aligned to the human organisational 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 (Hew and Cheung 2014).

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 behaviour and generically explains the joint venture organisational style. Section 4 describes the (requirements) scenarios that must be supported by the proposed MAS architecture thus software platform. Section 5 then instantiates the generic joint venture organisational style to an MOOC behaviour and

describes the main roles involved in it. Section 6 describes the design of the MAS to be set-up and its behaviour 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 Armbrust et al. 2010 for a definition and characterisation), the possible use of VLE as distributed architectures aggregating heterogeneous software systems to support learning processes has been identified. Indeed, Masud and Huang (2012) 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.

Ivanovic and Jain (2014) 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 modelling and design paradigm for advanced human–computer interaction and (ii) agents for smart functional decomposition of complex systems*. In the same book, Tibaut et al. (2014) nevertheless only develop the first aspect through the development of ICTEM, an inter-university collaboration VLE allowing to build a virtual learning community based on a common course pool. Their VLE requires a common inter-university taxonomy for information classification and organisation that all members of the virtual learning community agree upon. The common taxonomy allows one to build 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 Gillani and Ko (2015) . The taxonomy constitutes, following us, 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, Tibaut et al. (2014) only consider agents as real-life collaborating peers (so approach (i)) while we build and map this real-life organisation 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 behaviour of individual agents with the system behaviour at run-time. This trend is known, in literature, as social engineering (Yu et al. 2011). Within this discipline, patterns of organisational and social behaviour have been distinguished over the years. For instance, the joint-venture pattern was firstly applied to an MOOC architecture by Kolp and Wautelet (2015) but with no realisation 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 to dispose of a software system behaving following an experienced human organisational pattern to support processes involving a huge set of collaborating software agents. Gluz, Vicari, and Passerino (2015) emphasise the necessity to understand the social and cultural aspects of software analysis and design when modelling applications involving a lot of users out of various contexts; they notably illustrate it with the modelling 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 Pireva et al. (2014). 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 five key issues addressed by agents in the field of cloud e-learning systems in order to build an MOOC, i.e. *learner-centred* , *openness*, *personalisation* , *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 an MOOC present in the MOOC platform or communicate about the MOOC (e.g. Coerderoi and Vas 2015). 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. Ting et al. (2015) 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 this study 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 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. Gillani and Eynon (2014) showed that students that actively participate to discussions tend to better succeed on evaluations so that communication should be monitored and managed optimally. Hew and Cheung (2014) explain 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 fulfil* . In the same way, these authors highlight four 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 to 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 #hashtag function, etc.) 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 specialised 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 optimisation 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), optimisation of time devoted to the MOOC will give more time to work on activities such as 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 three 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 artefacts presented and realised 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 Yu et al. 2011) which offer actors (agents, roles or positions), goals, and actor dependencies as primitive concepts for modelling an application during early requirements or organisational /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 modelling the organisational 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 up using a field analysis, leading to a precise identification of requirements (see Section 4). To build this heterogeneous VLE that is able to 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.

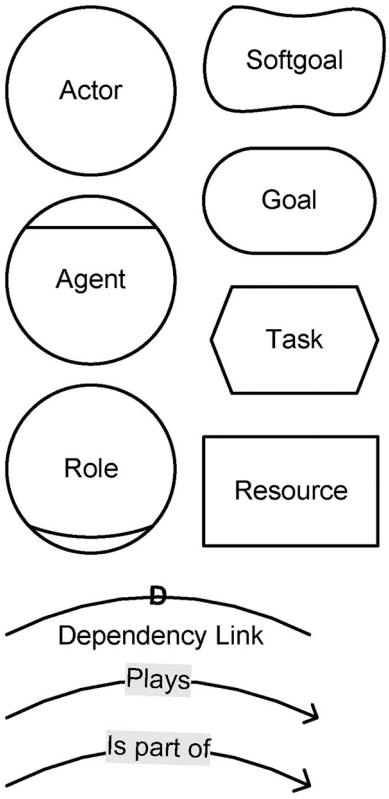

Figure 1. *i*∗ Elements and their graphical representation.

Pattern-based MAS development allows indeed to map (validated) human organisational behaviour that is generally applied when facing similar situations in real life to software system behaviour. As evoked earlier, in order to structure our MAS architecture, we use a defined organisational 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 .' (Morabito, Sack, and Bhate 1999). Organisation 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. Organisation theory describes how practical organisations 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 organisation theory have proposed models and patterns to try to find and formalise recurring organisational structures and behaviours.

Today, organisational structures are primarily studied by two disciplines:*Organisation Theory* (e.g. Daft 2012; Mintzberg 1992; Scott 1998; Yoshino and Rangan 1995), which describes the structure and design of an organisation; *Strategic Alliances* (e.g. Dussauge and Garrette 1999; Gomes-Casseres 1996; Morabito, Sack, and Bhate 1999; Segil 1996; Todeva and Knoke 2005), which model the strategic collaborations of independent organisational stakeholders who have agreed to pursue a set of agreed-upon business goals. Both disciplines aim to identify and study organisational patterns. These are not just modelling 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 behaviours. Rather, it exists in various forms at every stage of crystallisation (i.e. specification), and at every level of granularity in the organisation. The more manifest is its representation, the more the pattern emerges and becomes recognisable – whether at a high or low level of granularity.

Kolp and Wautelet (2015) propose a set of organisational styles for agent-oriented software engineering; the idea is to map patterns adopted in human organisational theory to the behaviour of MAS, thus to the internal behaviour 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 organisation, 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 an MOOC context is depicted in Kolp and Wautelet (2015). No realisation scenarios are nevertheless defined nor any design and implementation set in place. The structure is very relevant for building an MOOC platform as a 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 in 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 up an MAS architecture showing how MOOC organisational 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 (Kolp, Faulkner, and Wautelet 2008; Kolp, Wautelet, and Faulkner 2011). In other words, the design involves the description of the system agents' behaviour in order to fulfil the identified scenarios. The organisational 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 an MOOC platform. To this end, it depicts standard scenarios. Communication issues are vital for the success of an MOOC with potentially a huge amount of students and a very little support staff. This staff can impossibly reply all of the questions or furnish all the material requested so that the students' community itself has to help in the global effort. Indeed, most of the requests (questions, need for material, etc.) can be addressed by other students and little requests need to be escalated to junior (e.g. Teaching Assistants (TA)) or senior (e.g. professors) support staff. Such scenarios are envisaged and conceptualised 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 (i) to have access to learning material (e.g. documents, slides, books, videosnot yet provided or not accessible by the requester (e.g. because he does not have access to the system providing it), or (ii) to ask for a service concerning learning material (e.g. a question about the course, be evaluated on an exercise). For simplicity, we assume here that every time a request for a `Learning Object` is issued, the addressed `Learning Peer` can satisfy that request in three principal ways: (a) using its 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 fulfil 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 to the request so that it 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 minimises the *time to answer* but requires the knowledge or ability to answer the request; in a word, he has to own the*competence* to fulfil 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 other known`Learning Peers` (at an upper rank in their internal organisation scale) that 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, it 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 fulfil the requests; it 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 their own decisions about how to answer requests at best based on the experience and their pedagogical approach. This, however, on the condition that 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 an 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 (pre-defined) teaching guidelines that could constitute a drawback to the flexibility in the request satisfaction.

Finally, in scenario (c), it could be that an `Instructor` has the competence to answer the request but that he does not have the time to fulfil 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, etc.). In a sentence, the `Learning Peer` delegates the request to a`Learning Peer` (community) at the same level or lower 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 an MOOC. While in an independent[2] scenario (a) the request is fulfilled optimally (minimising 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 the lowest level possible so that only the ones requiring very specific competencies are escalated to the (costly) highest levels; this is however not optimal in terms of time to answer (so that a better compromise can possibly be found).

5. Organisation-based architectural design

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

5.1. Organisational 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 organisations composed of autonomous and proactive agents that cooperate with each other to achieve common or private goals (Hu et al. 2014; Mao and Yu 2004; Mylopoulos, Kolp, and Giorgini 2002). A key aspect to conduct macro-level architectural design is the specification and use of organisational styles (Kolp and Wautelet 2015). These are social-based design alternatives inspired by models and concepts from organisational theories that analyse the structure and design of real-world human organisations (see, for example, Daft 2012). 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 organisational 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 behaviour 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 decentralised organisation to better fit a platform serving as an abstraction layer over other software environments that has to support the requirements

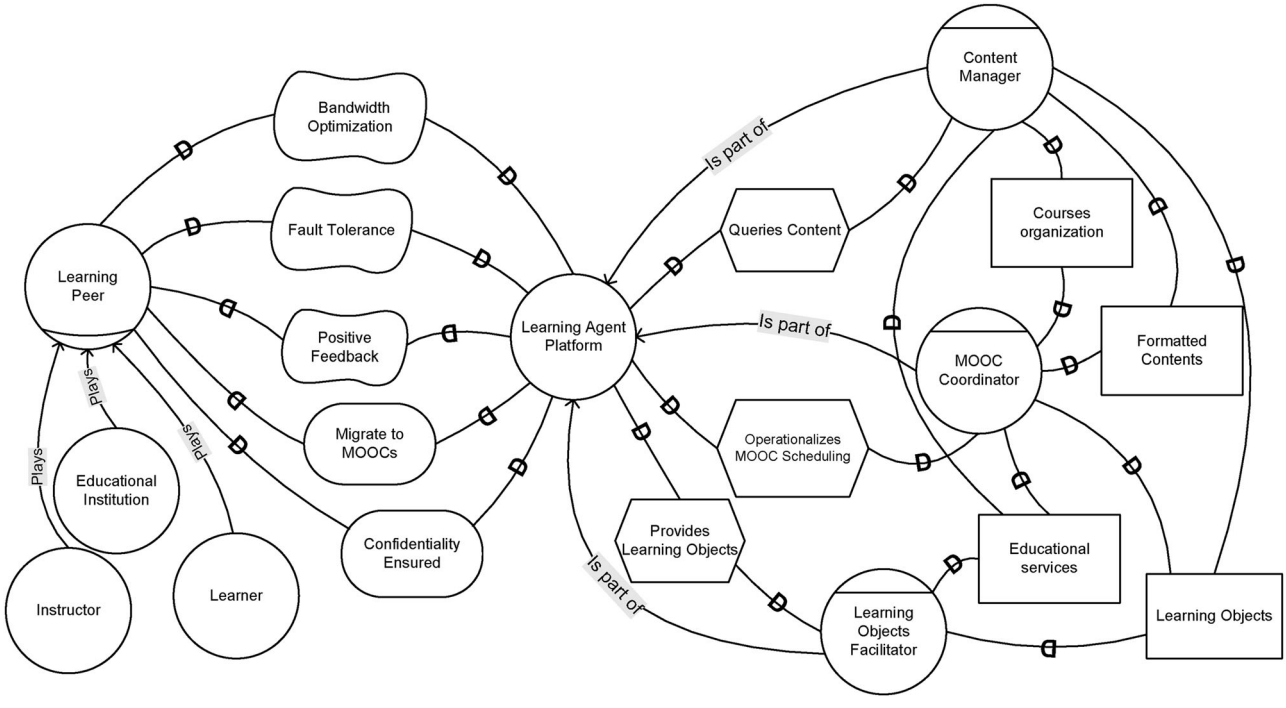

Figure 2. An MOOC in ajoint venture style.

depicted in Section 4. The principal partners of the joint venture are autonomous actors (or agents and roles) that directly exchange services, data; and knowledge with each other similarly to the partners composing a network of decentralised learning agents in an MOOC context (see, e.g. Kop 2011). Figure 2 shows that there are several (real-life) actors, i.e. Instructor, Educational Institution (e.g. university and high school) and Learner, that can play the online avatar role of Learning Peer which 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 online teaching) to MOOCs. Similarly, the Learning Agent Platform is depending on the Learning Peer for having Confidentiality Ensured as well as software quality attributes such as Bandwidth Optimisation 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 fulfil 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 Manager Queries Content, the MOOC Coordinator Operationalises MOOC Scheduling and the Learning Object Facilitator Provides 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 Organisation to theContent Manager as well as Educational Services to the Learning Object Facilitator. Finally, the Learning Object Facilitator provides Learning Objects resources both to the Content Manager and the MOOC Coordinator.

5.2. System agents and organisational 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 both requirements and organisational styles analysis, in order to satisfy the (business) organisational 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 realise. Such agents will be provided by the software system and supported by means of agent abilities.

At this modelling level, even if the business organisational roles played by the organisational 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 organisational 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/organisational peer:

The *Learning Objects Facilitator (searching and registration)*. As emerged 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 Objects 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, watching at Figure 2, each agent inside the Learning Agent Platform depends on the Learning Object Facilitator to perform the taskProvide 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 provide 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

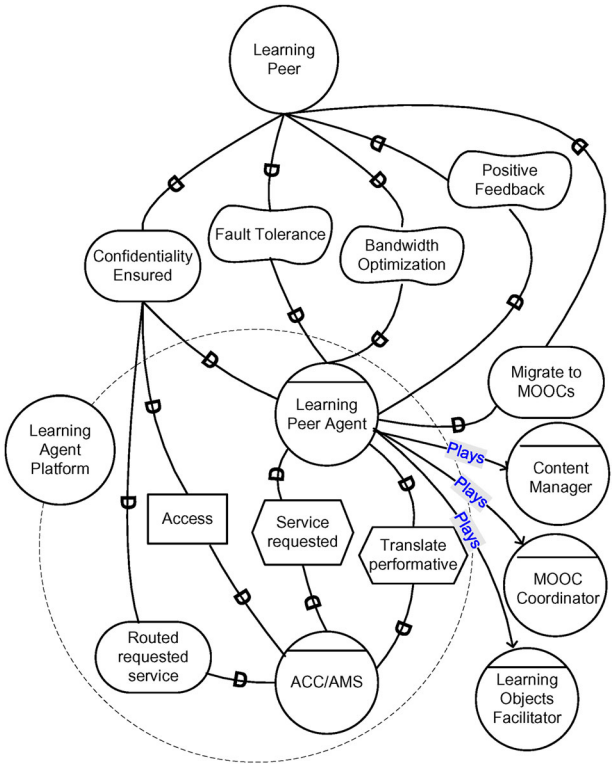

Figure 3. The `Learning Agent Platform` as an embassy agent pattern.

cannot be internally satisfied, the `MOOC Coordinator` or the `Content Manager` could interact with the `Learning Objects Facilitator` agent to get new acquaintances. Specifically, such a behaviour 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 platforms facilitators. Such a behaviour 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), it needs to interact with different software systems or platforms used by various `Learning Peers`. Requests could indeed be issued from a CMS, an 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 well-known 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 (see, e.g. Panti, Spalazzi, and Penserini 2001). 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 decentralised`Learning Peers`, i.e. virtual learning members. For instance, when a failure results from inability of the `Learning Agent Platform` to satisfy a request locally, the `MOOC Coordinator` can help one to build up a cooperation strategy in order to overcome the underling 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 operationalise the MOOC scheduling, i.e.*ensure requests fulfilment* , *to query information sources*, *to edict guidelines to be followed*, etc. To this end, such an agent can rely on the well-known BDI architecture (Bratman 1999; Casali, Godo, and Sierra 2011; Pokahr et al. 2014; Rao and Georgeff 1995; Wautelet and Kolp 2016). 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, it chooses the more convenient behaviour (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 update all the learning repositories. 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), for example, repositories to get the status of specific `Learning Objects`. To this end, Figures 2 and 4 indicate such relations by means of resource

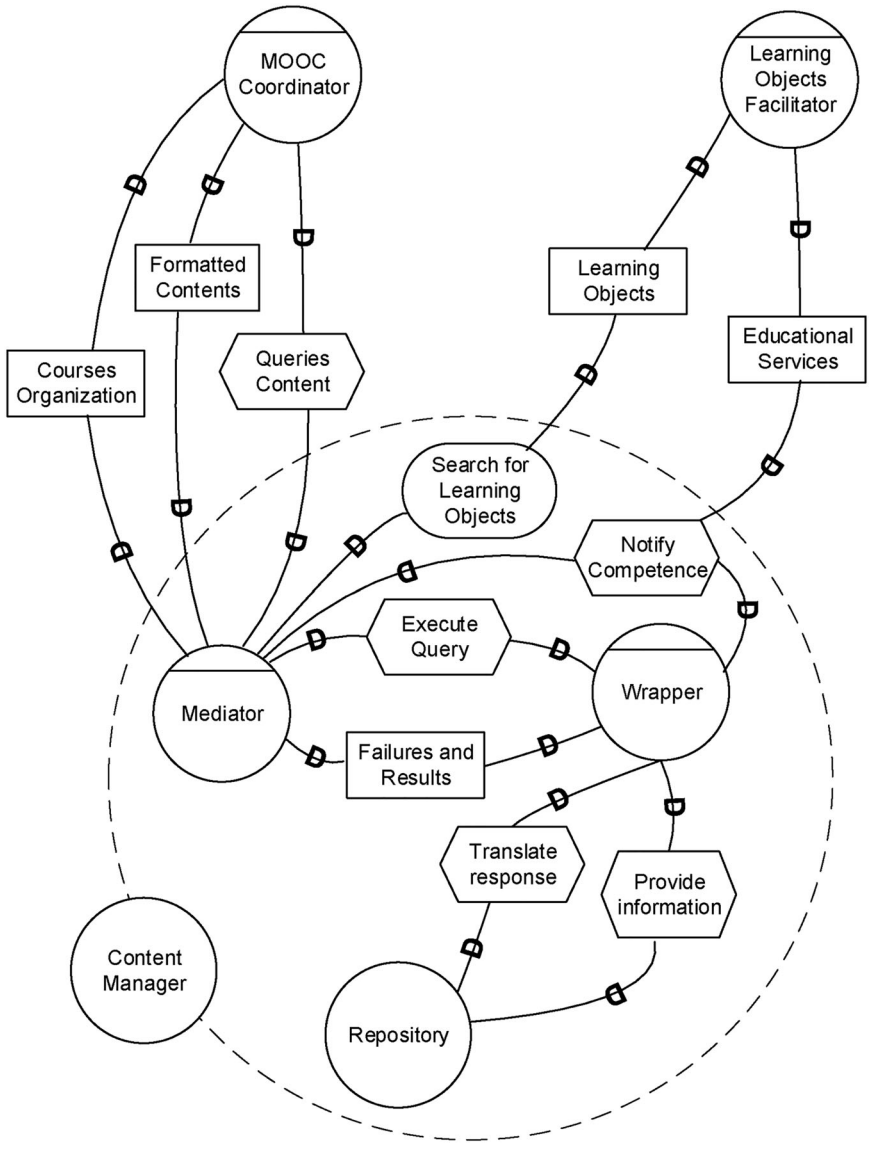

Figure 4. The `Content Manager` as a mediator agent pattern.

dependencies `Formatted Contents` and `Courses Organisation`.

6. Detailed design of the MOOC architecture as a joint venture

The joint venture style applied to the MOOC architecture in Section 5 defines the logical behaviour of the system-to-be developed. Indeed, every time an organisational style is applied to the system designer it allows to easily point up the organisational actors and roles that need to be supported by the system. The following step in the MAS development thus requires to map and detail the system agents and organisational roles to specific software agents and characterise their behaviour. Namely, each role in Figure 2 is much closer to the real organisational actor behaviours than software agent behaviours that we aim to achieve. As a consequence, the organisational characterisation that we have made allowedus to determine 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/modelling 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 are focusing on social and intentional aspects that are recurrent in MAS or cooperative systems. Moreover, similarly to organisational styles, social patterns are generic structures that define how (a small number of) agents are interacting together in order to fulfil their obligations. Social patterns are applied here to depict the system behaviour ;

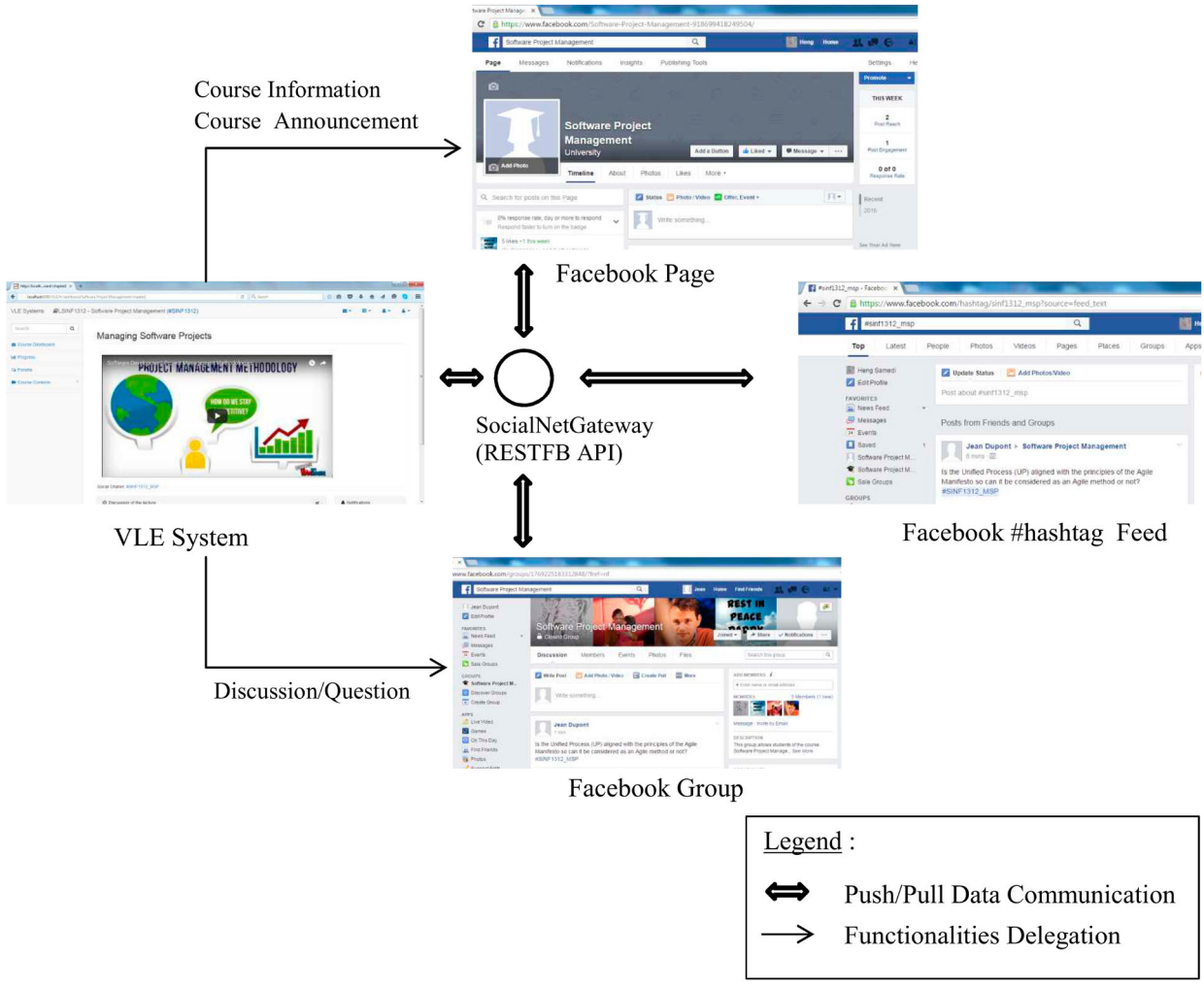

Figure 5. Mechanisms used into Facebook to (re)present the MOOC.

roughly speaking they map the organisational behaviour 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) characterising each system actor in terms of agents and their interdependencies needed to accomplish the organisational roles; (b) organising such agents in recurrent structures, named social patterns, in order to allow designers for the reuse of design experience and knowledge during the whole system development process. According to (a) and (b), we modelled each organisational 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`.

In Figure 3, the embassy agent pattern (Kolp, Faulkner, and Wautelet 2008; Kolp, Wautelet, and Faulkner 2011) has been applied to build up 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 responsible for all the tasks delegated by the corresponding `Learning Peer` (see Penserini et al. 2003 for details). Figure 3 explicitly shows the `PLAYS` link dependencies. A`Learning Peer` depends on the agent `ACC/AMS` to submit messages and to gain access to the agent platform resources, as follows. Each `Learning Peer`

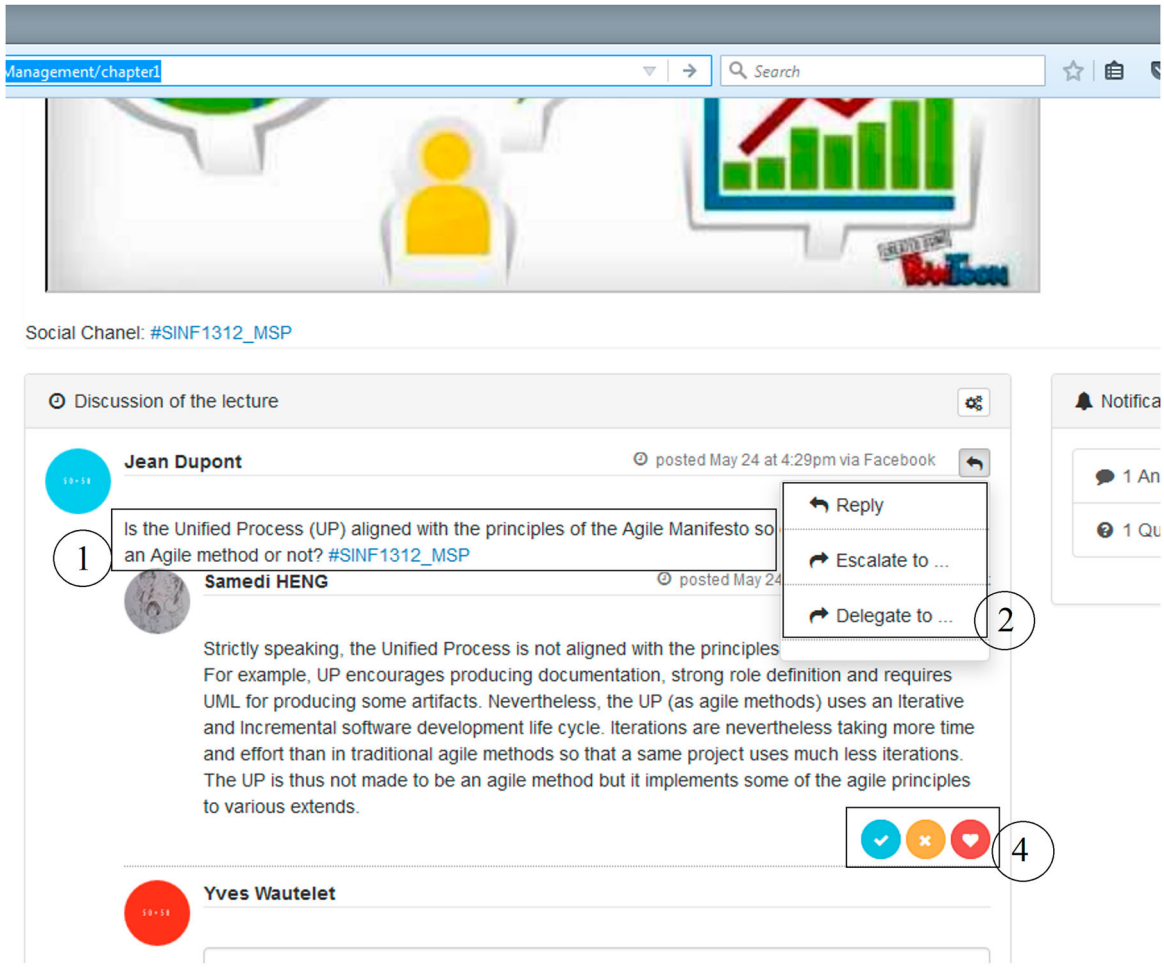

Figure 6. Instructor view of a request for learning objects in the native VLE.

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 to 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).

Figure 4 gives another example of how to model organisational roles by means of agent patterns. Specifically, the mediator agent pattern (Kolp, Faulkner, and Wautelet 2008; Kolp, Wautelet, and Faulkner 2011) is adopted to correctly match the `Content Manager` requirements arose from the previous (social-based) organisational 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 semi-structured data (see, e.g. Papakonstantinou, Garcia-Molina, and Ullman 1996; Wiederhold 1992). Into our system, the`Mediator` plays a key role because it 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`, it 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, it 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

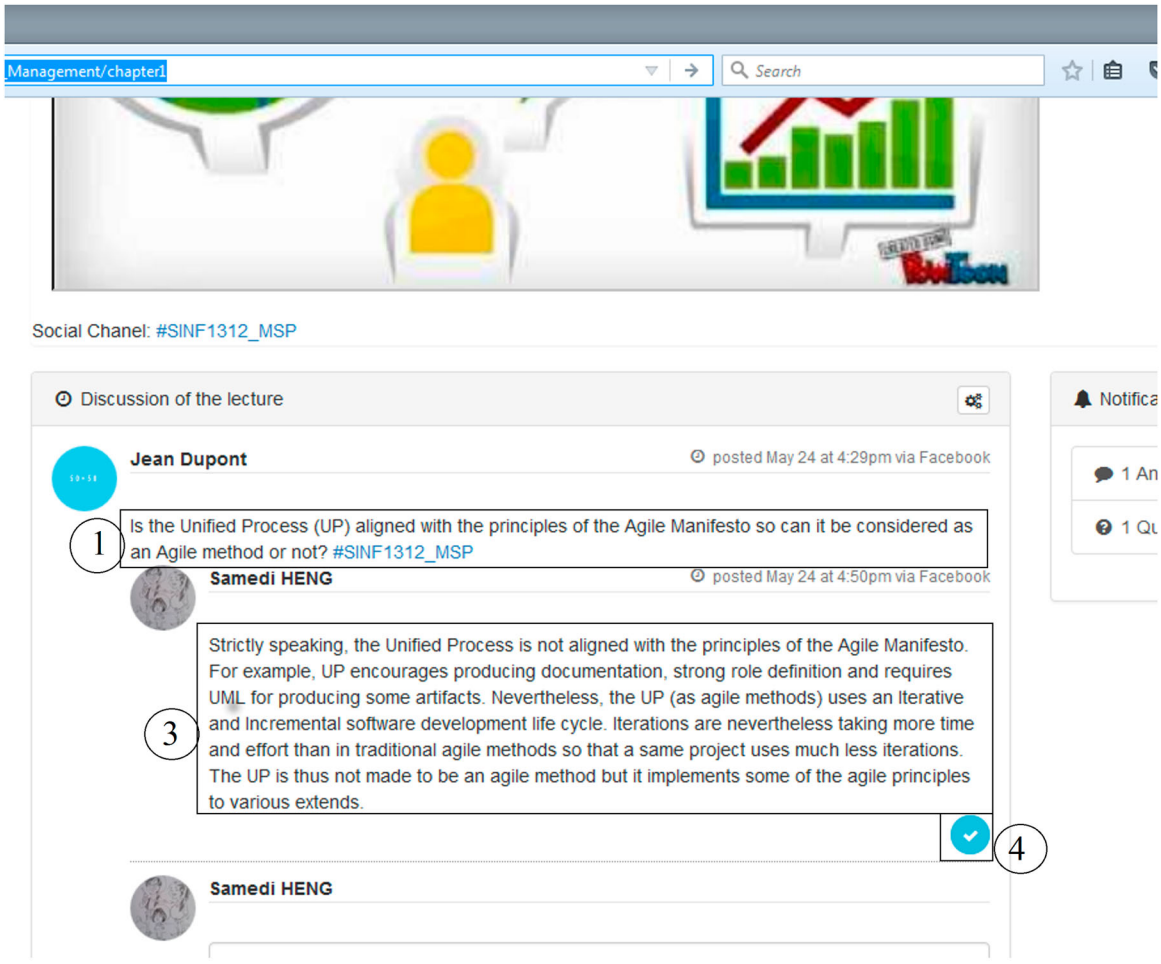

Figure 7. Student view of a request for learning objects in the native VLE.

`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 fulfil 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 Figures 2 and 3, each `Content Manager` can interact with different`Learning Peers`, specifically peers that 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 depicted in Appendix 1, the dialogue between the GUI and the MAS are overviewed in Appendix 2.

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 synchronised at the level of the MAS platform.

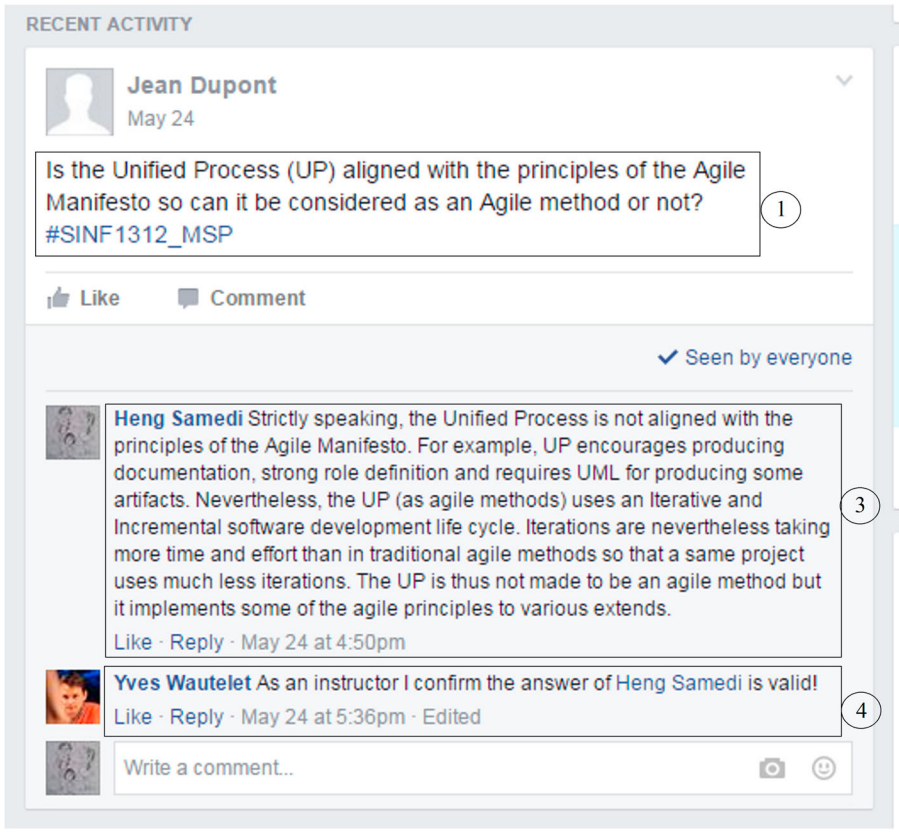

Figure 8. Learning peer view of a request for learning objects in the Facebook.

When an 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, it also automatically joins the Facebook page and gets notified of updated content through the Facebook notification system.`Learning Peer`s can interact with other `Learning Peer`s 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 centralised in a Facebook group for the MOOC (also generated when the MOOC is created inthe Facebook); communication is detailed in the next section. All posts and comments on the Facebook page are systematically synchronised with the main Facebook course page and with the native MOOC platform.

7.2. An interface for supporting the requirements scenarios

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

Figures 6–8 show that 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 fulfil the request. Box (3) existing in Figures 7 and 8 allows one to visualise 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 Figures 7 and 8, the Instructor can validate an answer; this is done in Facebook by an instructor using the *@function*.

8. Conclusion

The evolution of e-learning activities has lead to the creation of VLE with more and more functions offering real

perspectives for effective distance learning. As an ultimate development, we find MOOCs that can encompass thousands of learners needing to be actively supported by instructors. Indeed, without proper support, an 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 on average only 10–20 % 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 a heterogeneous software environment to instructors. We indeed suggest to build an MOOC that works as an upper layer to classical VLE allowing one to bring 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 to deal with the complexity of communications in heterogeneous software environments in a manner that various companies set up a joint governance. Software agents – playing the roles found in the joint venture and instantiated to the MOOC context – then behave/organise as real-life entities.

The paper showed the feasibility of the proposal by developing a usable prototype; its interaction with Facebook allows one to synchronise 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 to 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, an MOOC is currently under development in the research department. When the MOOC will be deployed, we will perform an in-depth evaluation of the perceived user experience by students and instructors.

Notes

1. Multiple definitions of Learning Objects exist in the literature, Churchill (2007) defines 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 conceptualisation, the `Learning Object` can be considered as a modular resource that can be digitised (so dematerialised) to be used and re-used for the support of learning (and e-learning) activities. Further Examples are provided in the section.
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.
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.
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 Balasubramanee et al. 2013.
7. http://restfb.com/
8. https://developers.facebook.com/docs/graph-api

Disclosure statement

No potential conflict of interest was reported by the authors.

ORCiD

Yves Wautelet http://orcid.org/0000-0002-6560-9787

References

Armbrust, M., I. Stoica, M. Zaharia, A. Fox, R. Griffith, A. D. Joseph, and R. Katz, et al. 2010. "A View of Cloud Computing." *Commun. ACM* 53 (4): 50–58.

Balasubramanee, V., C. Wimalasena, R. Singh, and M. Pierce. 2013. "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*, Vol. 1. IEEE Computer Society.

Bellifemine, F., G. Caire, and D. Greenwood. 2007. *Developing Multi-Agent Systems with JADE, Vol. 5*. Chichester: Wiley.

Bratman, M. E. 1999. *Intention, Plans, and Practical Reason*. Cambridge: Cambridge University Press.

Casali, A., L. Godo, and C. Sierra. 2011. "A Graded BDI Agent Model to Represent and Reason About Preferences." *Artificial Intelligence* 175 (7–8): 1468–1478.

Churchill, D. 2007. "Towards a Useful Classification of Learning Objects." *Educational Technology Research and Development* 55 (5): 479–497.

Coerderoi, R., and A. Vas. 2015. MOOC Fondements de la stratégie d'entreprise. [online] https://www.facebook.com/moocstrat/timeline

Crockford, D. 2006. The application/json Media Type for JavaScript Object Notation (JSON). [online], (4627). https://rfc-editor.org/rfc/rfc4627.txt

Daft, R. 2012. *Organization Theory and Design*. Mason: Cengage Learning.

Darwin, P.B., and P. Kozlowski. 2013. *AngularJS Web Application Development*. Birmingham: Packt Publishing.

Dillenbourg, P., A. Fox, C. Kirchner, J. C. Mitchell, and M. Wirsing. 2014. "Massive Open Online Courses: Current State and Perspectives (Dagstuhl Perspectives Workshop 14112)." *Dagstuhl Manifestos* 4 (1): 1–27.

Dillenbourg, P., D. Schneider, and P. Synteta. 2002. "Virtual Learning Environments." In *3rd Hellenic Conference "Information & Communication Technologies in Education"*, edited by P. Dillenbourg, A. Fox, C. Kirchner, J. Mitchell and M. Wirsing, 3–18. Dagstuhl: Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

Dussauge, P., and B. Garrette. 1999. *Cooperative Strategy: Competing Successfully Through Strategic Alliances*. Chichester: Wiley and Sons.

Flanagan, D. 2011. *JavaScript: The Definitive Guide Activate Your Web Pages*. 6th ed. Sebastopol: O'Reilly Media, Inc.

Fong, W.W. 2011. "From Web 2.0 to Classroom 3.0." In *Hybrid Learning – 4th International Conference, ICHL 2011, Hong Kong, China, August 10–12, 2011. Proceedings*, edited by R. Kwan, J. Fong, L. F. Kwok and J. Lam, 298–305. Vol. 6837 of *Lecture Notes in Computer Science*, Heidelberg: Springer.

Fraternali, P., G. Rossi, and F. Sánchez-Figueroa. 2010. "Rich Internet Applications." *IEEE Internet Computing* 14 (3): 9–12.

Gillani, N., and R. Eynon. 2014. "Communication Patterns in Massively Open Online Courses." *The Internet and Higher Education* 23: 18–26.

Gillani, S. A., and A. Ko. 2015. "Incremental Ontology Population and Enrichment Through Semantic-based Text Mining: An Application for IT Audit Domain." *Int. J. Semantic Web Inf. Syst.* 11 (3): 44–66.

Gluz, J., R.M. Vicari, and L. Passerino. 2015. "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*, Vol. 606 of *Communications in Computer and Information Science*, edited by F. Koch, A. Koster and T. Primo. Heidelberg: Springer.

Goldstein, A., L. Lazaris, and E. Weyl. 2015. *HTML5 & CSS3 for the Real World*. Collingwood: sitepoint.

Gomes-Casseres, B. 1996. *The Alliance Revolution: The New Shape of Business Rivalry*. Cambridge, MA: Harvard University Press.

Hew, K. F., and W. S. Cheung. 2014. "Students' and Instructors' Use of Massive Open Online Courses (MOOCs): Motivations and Challenges." *Educational Research Review* 12: 45–58.

Hu, C., X. Mao, M. Li, and Z. Zhu. 2014. "Organization-Based Agent-Oriented Programming: Model, Mechanisms, and Language." *Frontiers of Computer Science* 8 (1): 33–51.

Ivanovic, M., and L. C. Jain, eds. 2014. *E-Learning Paradigms and Applications - Agent-based Approach, Vol. 528*, Studies in Computational Intelligence. Heidelberg: Springer.

Kolp, M., S. Faulkner, and Y. Wautelet. 2008. "Social Structure Based Design Patterns for Agent-Oriented Software Engineering." *IJIIT* 4 (2): 1–23.

Kolp, M., and Y. Wautelet. 2015. "Human Organizational Patterns Applied to Collaborative Learning Software Systems." *Computers in Human Behavior* 51: 742–751.

Kolp, M., Y. Wautelet, and S. Faulkner. 2011. "Social-Centric Design of Multi-Agent Architectures." In *Social Modeling for Requirements Engineering*, edited by P. Giorgini, N. Maiden, J. Mylopoulos and E. Yu. Cambridge: MIT Press.

Kop, R. 2011. "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.

Lytras, M. D., H. I. Mathkour, H. Abdalla, W. Al-Halabi, C. Yanez-Marquez, and S. W. M. Siqueira. 2015. "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 (0): 557–561.

Mao, X., and E. S. K. Yu. 2004. "Organizational and Social Concepts in Agent Oriented Software Engineering." In *Agent-Oriented Software Engineering V, 5th International Workshop, AOSE 2004, New York, NY, USA, July 19, 2004, Revised Selected Papers*, edited by J. Odell, P. Giorgini and J. P. Muller, 1–15. Vol. 3382 of *Lecture Notes in Computer Science*. Heidelberg: Springer.

Masud, M. A. H., and X. Huang. 2012. "An e-Learning System Architecture based on Cloud Computing." *System* 10 (11): 255–259.

Mesbah, A., and A. van Deursen. 2007. "Migrating Multi-page Web Applications to Single-page AJAX Interfaces." In *Proceedings of the 11th European Conference on Software Maintenance and Reengineering (CSMR '07)*, 181–190. Washington, DC: IEEE Computer Society.

Mikowski, M., and J. Powell. 2013. *Single Page Web Applications: JavaScript End-to-end* (1st ed.). Greenwich, CT: Manning Publications Co.

Mintzberg, H. 1992. *Structure in Fives : Designing Effective Organizations*. London: Prentice-Hall.

Morabito, J., I. Sack, and A. Bhate. 1999. *Organization Modeling : Innovative Architectures for the 21st Century*. London: Prentice-Hall.

Mylopoulos, J., M. Kolp, and P. Giorgini. 2002. "Agent-Oriented Software Development." In *Methods and Applications of Artificial Intelligence, Second Hellenic Conference on AI, SETN 2002. Thessaloniki, Greece, April 11–12, 2002, Proceedings*, edited by I. P. Vlahavas and C. D. Spyropoulos, 3–17. Vol. 2308 of *Lecture Notes in Computer Science*. Heidelberg: Springer.

Panti, M., L. Spalazzi, and L. Penserini. 2001. "Cooperation Strategies for Information Integration." In *Cooperative Information Systems, 9th International Conference, CoopIS 2001, Trento, Italy, September 5–7, 2001, Proceedings*, edited by C. Batini, F. Giunchiglia, P. Giorgini and M. Mecella, 123–134. Vol. 2172 of *Lecture Notes in Computer Science*. Heidelberg: Springer.

Papakonstantinou, Y., H. Garcia-Molina, and J. D. Ullman. 1996. "MedMaker: A Mediation System Based on Declarative Specifications." In *Proceedings of the Twelfth International Conference on Data Engineering, February 26–March 1, 1996, New Orleans, Louisiana*, edited by S. Y. W. Su, 132–141. Washington, DC: IEEE Computer Society.

Penserini, L., L. Liu, J. Mylopoulos, M. Panti, and L. Spalazzi. 2003. "Cooperation Strategies for Agent-Based P2P Systems." *Web Intelligence and Agent Systems* 1 (1): 3–21.

Pireva, K., P. Kefalas, D. Dranidis, T. Hatziapostolou, and A. Cowling. 2014. "Cloud e-Learning: A New Challenge for Multi-Agent Systems." *Agent and Multi-Agent Systems: Technologies and Applications*, 277–287. Heidelberg: Springer.

Pokahr, A., L. Braubach, C. Haubeck, and J. Ladiges. 2014. "Programming BDI Agents with Pure Java." In *Multiagent System Technologies – 12th German Conference, MATES 2014, Stuttgart, Germany, September 23–25, 2014. Proceedings*, edited by J. P. Müller, M. Weyrich and A. L. C. Bazzan, 216–233. Vol. 8732 of *Lecture Notes in Computer Science*, Springer.

Poslad, S. 2007. "Specifying Protocols for Multi-Agent Systems Interaction." *TAAS* 2 (4): 15:1–15:24.

Rao, A. S., and M. P. Georgeff. 1995. "BDI Agents: From Theory to Practice." In *Proceedings of the First International Conference on Multiagent Systems, June 12–14, 1995, San Francisco, CA, USA*, edited by V. R. Lesser and L. Gasser, 312–319. Cambridge: The MIT Press.

Scott, W. 1998. *Organizations: Rational, Natural, and Open Systems*. Chichester: Prentice-Hall.

Segil, L. 1996. *Intelligent Business Alliances : How to Profit Using Today's Most Important Strategic Tool*. New York: Times Business.

Tibaut, A., D. Rebolj, K. Menzel and R. Jardim-Goncalves. 2014. *Inter-university Virtual Learning Environment. Ivanovic and Jain (2014)*. 97–119. Heidelberg: Springer.

Ting, I-H., W-J. Wu, H-T. Kao, and D. Wang. 2015. An Implementation of Online Learning and Course Management System Based on Facebook, *Learning Technology for Education in Cloud*, Communications in Computer and Information Science, Vol. 533, 208–218. Heidelberg: Springer.

Todeva, E., and D. Knoke. 2005. "Strategic Alliances and Models of Collaboration." *Management Decision* 43 (1): 123–148.

Wautelet, Y., and M. Kolp. 2016. "Business and Model-Driven Development of BDI Multi-Agent Systems." *Neurocomputing* 182: 304–321.

Weller, M. 2007. *Virtual Learning Environments: Using, Choosing and Developing your VLE*. New York: Routledge.

Wiederhold, G. 1992. "Mediators in the Architecture of Future Information Systems." *IEEE Computer* 25 (3): 38–49.

Wulf, J., I. Blohm, J. M. Leimeister, and W. Brenner. 2014. "Massive Open Online Courses." *Business & Information Systems Engineering* 6 (2): 111–114.

Yoshino, M., and U.S. Rangan. 1995. *Strategic Alliances : An Entrepreneurial Approach to Globalization*. Boston: Harvard Business School Press.

Yu, E., P. Giorgini, N. Maiden, and J. Mylopoulos. 2011. *Social Modeling for Requirements Engineering*. Cambridge: MIT Press.

Appendix 1. Technologies and programming languages used

For the development of the GUI, the *AngularJS* framework[4] (Darwin and Kozlowski 2013) has been used in combination with *Bootstrap*[5] (Balasubramanee et al. 2013). AngularJS is a *Model-View-Controller* (*MVC*) framework based on the *JavaScript* language (Flanagan 2011) that facilitates the development and the maintenance of single page (see Mesbah and van Deursen 2007; Mikowski and Powell 2013) and rich (see Fraternali, Rossi, and Sánchez-Figueroa 2010) internet applications. AngularJS uses extra tags embedded within HTML pages (Goldstein, Lazaris, and Weyl 2015) for showing 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 (Crockford 2006). 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 (Goldstein, Lazaris, and Weyl 2015) 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 it does not provide the JavaScript library, allowing one to build to rich and single-page applications (as AngularJS does).

For the server side, the MAS has been implemented using the *Java Agent DEvelopment Framework* (*JADE*) (Bellifemine, Caire, and Greenwood 2007). JADE is a framework used for implementing MAS which conforms to the FIPA standard (see Poslad 2007). 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 for implementing agents and uses `ACLMessages` (see Bellifemine, Caire, and Greenwood 2007) for communicating between agents. Jade can nevertheless not directly communicate with a web-browser by the use of the HTTP protocol but provides a gateway for communicating 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.

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 for pushing and pulling 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 unfortunately not work: Facebook does not provide any push mechanism for an external platform. Therefore, the SocialNetGateway sets a timer for pulling information from Facebook and pushing the update to the native MOOC platform. As a consequence, an update in Facebook is not instantly synchronised with native MOOC platform; the update time depends on the value of the timer set in SocialNetGateway.

Appendix 2. 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 such as 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.

Destination	Source	Operation	Content

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

Figure A.1. Client request in JSON format.

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 it can 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 A1. 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 `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 A1.

As mentioned earlier, the goal of the MAS architecture is to allow aggregating content and `Learning Peer` communities from heterogeneous software platforms such as social networks to enrol 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 such as Facebook, whereas `MOOCGateway` is in charge of exchanging course content with other VLE. Figure A2 exposes the architecture of our MAS platform communicating with other software environments.

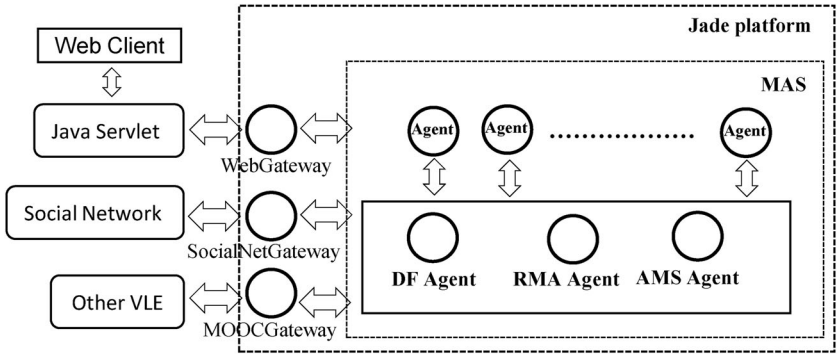

Figure A.2. MOOC deployment architecture for integration of heterogeneous platforms.