Model-Driven Generation of Multi-User and Multi-Domain Choreographies for staging in Multiple Virtual World Platforms

Emanuel Silva[1,2], Nuno Silva[1], Leonel Morgado[3]

[1]School of Engineering, Polytechnic of Porto, Porto, Portugal
[2]University of Trás-os-Montes e Alto Douro, Vila Real, Portugal
{ecs, nps}@isep.ipp.pt
[3]INESC TEC (formerly INESC Porto) / Universidade Aberta, Lisbon, Portugal
leonel.morgado@uab.pt

Abstract. This paper presents an approach that enables the staging of choreographies for education and training purposes in multiple virtual world platforms. Choreography is the description of a set of actions that must or may be executed by a group of participants, including the goals to be achieved and any restrictions that may exist. For capturing and representing multi-actor multi-domain choreographies an approach based on ontologies with distinct levels of abstraction is adopted. Further, this paper proposes a modelling driven approach and a set of processes that, through mappings between ontologies, enable the automatic construction of a platform-specific choreography from a platform-independent one, thus reducing the time and effort of the choreography development. For this, the MDA paradigm was adopted and adapted in a way where models can reflect two dimensions of independence: platform independence and application domain independence. We also point the guidelines for staging the choreography in a virtual world platform.

Keywords: virtual worlds, training, choreography, multi-user, model-driven, ontology, mapping, transformation

1 Introduction

Virtual worlds have achieved significant levels of interest for supporting teaching and learning activities [1], [2] since they provide educators with tools for creation of immersive environments where multiple characters together can develop skills in a simulated context sharing a common virtual space [3], [4]. Choreographies of virtual actors are a specific type of content that represent the set of actions that can be performed simultaneously by human-users and virtual computer-controlled actors thus enabling human trainees/students to play roles as part of teams or within a simulated social context. In this sense, a choreography is the description of a set of actions that must or may be executed by a group of participants, including the goals to be achieved and any restrictions that may exist.

Designing a choreography is a resource-intensive effort so, it is important that the result will not stay tied to a specific virtual world platform (VWP) but available to be applied on any VWP. However, as virtual platforms are very heterogeneous in terms of (e.g.) functionalities, data models, execution engines and programming/scripting languages or APIs, deploying a platform-based choreography into another VWP is difficult and time-consuming [5]–[8].

In this paper we present an approach that provides a contribution that facilitates the development, sharing and generation of choreographies aimed to be staged in different virtual platforms. For this, we suggest an approach where the conceptual representation model of the choreography is captured in the form of ontologies, and its adaptation to a particular virtual world follows a set of model transformation processes, similar to that suggested by the Model Driven Architecture (MDA) paradigm [9]. The proposed ontology-based definition of choreographies can capture not only the physical aspects as objects but more complex content such as procedures, consisting of sets of actions and conditions in which the actors can perform them.

Thus, this paper presents an approach that deals with the representation of platform-independent multi-user choreographies, and their deployment to different VWPs using a set of transformation processes based on ontologies and alignments. The rest of the paper comprehends six more sections. In section 2 the work related with our ideas is described. In section 3 we present the background knowledge necessary to understand better the transformation process. In Section 4 is described how to transform a choreography to a specific VWP and the corresponding algorithm is presented. Further, a validation of the choreography's feasibility is proposed. In section 5 aspects related to the execution of the choreography are described. Section 6 describes the developed experiments. Finally, Section 7 summarizes our ideas and suggests future directions.

2 Related Work

In the literature there is relevant work addressing the description of plans to represent training procedures to be staged by a single actor or by several elements, and how actions are distributed among them. However, most approaches design a choreography to be staged on a particular VWP. This creates strong dependencies with this VWP, making it difficult or even impossible to apply to other virtual worlds. Thus, related work can be categorized according to three dimensions: modeling independence, VWP independence and number and type of the actors.

Some approaches use separate models to represent the specification of procedures and scene [7], [10], [11]. They address team training scenarios but they are strongly dependent on the characteristics of the VWP. Some other approaches attempt to bridge the gap between the representation of procedures and its execution in distinct VWP. However, such approaches are only focused on a single user not allowing the representation of teamwork [12]–[14].

The representation of a choreography is very similar to what the W3C working group calls a task model (e.g. EOFM, AMBOSS, CTT, GOMS) [15]. These approaches propose hierarchical tree models in which tasks are successively decomposed into subtasks until an atomic level, where tasks represent actions that can be performed directly.

We intend to explore the ability to generate actions plan (using existing planners) for team members controlled by the computer, so we adopt a more simplified task model with a STRIPS-like representation [16] instead of a Hierarchical Task. Regarding the hierarchical models we can compare them to our STRIPS-like approach as follows: a choreography represents one task composed of a set of actions that represent subtasks. Thus, this model can be seen as a hierarchical structure with only one depth level.

Theoretically, models with hierarchical representation have a higher expressiveness than STRIPS. However, hierarchical models have severe limitations in their practical application since they can become undecidable, even under strong constraints and therefore being computationally unsuitable due to excessive computation time required to calculate action plans. In practice, restrictions are applied to restrict the size of the plan to make it finite, reduce the depth of the hierarchy (the number of levels of subtasks) and set order among tasks in order to establish a sequence between them.

By applying this set of constraints, the hierarchical model becomes very close and can even be expressed in STRIPS. So, it can be considered that in practice the expressivity of STRIPS and hierarchical models are similar [17].

We try bringing together all these features and present an approach capable of representing teamwork choreographies involving multi-users, capturing conceptually in a unique ontology model the actions and scene. We present also how to adapt the choreography to potentially any VWP and how to stage it.

3 Background Knowledge

To deal with VWP with different characteristics, we argue that choreographies should be clearly separated from the technical characteristics of the execution in the VWP [18]. To represent choreographies, we adopt a model based on ontologies instead of MOF/OCL metamodels for the following main reasons:

- ontologies are possibly the best way to represent conceptual information in order to bring the intelligent systems closer to the human conceptual level [19];
- ontologies allow different ways of describing classes. Can be defined which are the necessary conditions, i.e., conditions that must be met for an instance belonging to this class, and the necessary and sufficient conditions, i.e., the conditions that once fulfilled make an instance belongs to this class (cf. example of section 3.2);
- ontologies provide additional constructs like transitive closure for properties or represent disjoint classes;
- ontologies enable and automated reasoning, namely the dynamic classification of instances based upon class descriptions and automatic consistency checking when operated with a reasoner (cf. example of section 3.2).

Hence, the core of the proposal is a “generic high-level ontology” that captures the choreography in a conceptual and abstract model, so it is independent from the staging/deployment VWP. The data model of every virtual world must be captured by the so-called “platform-specific ontology”, and a mapping must be defined between these two ontologies. The mapping will provide the means to transform the platform-independent choreography into a platform-dependent one.

To address this, the MDA software-development paradigm [9] was adopted and adapted. Based on the concept of model independence and model transformation of MDA, we adopt an approach based on two first-class citizen dimensions: the VWP dimension and the choreography’s domain dimension. In fact, unlike in MDA, in our approach the model is not only independent from the VWP but also independent from the (choreography’s) domain. **Fig. 1** depicts the MDA one-dimensional approach (**Fig. 1** a) in comparison with the two-dimensional envisaged approach (**Fig. 1** b).

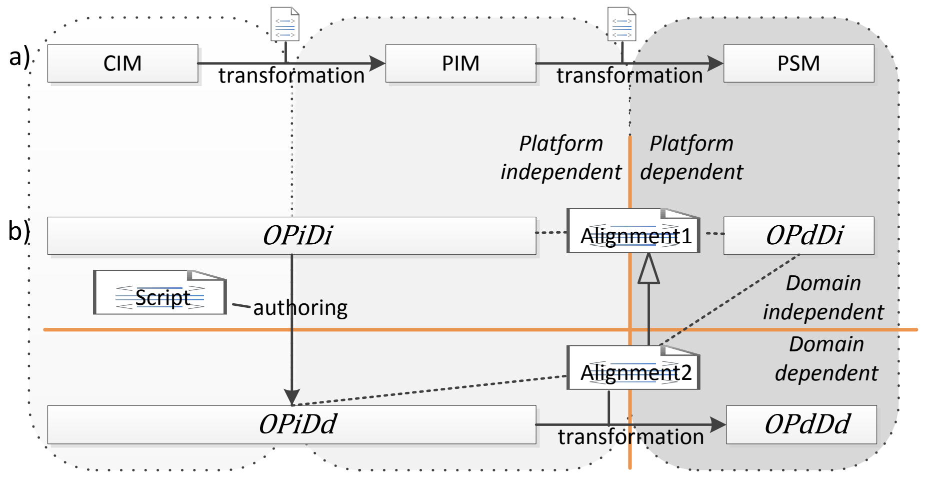


Fig. 1. – Model-Driven representation according to: a) MDA and b) Our approach

The nomenclature O_{PxDx} refers to Ontology, Platform and Domain, with “x” assuming “d” and “i” values for “dependent” and “independent”, respectively.

Taking into account the characteristics of ontologies described earlier, we claim that ontologies are then the adequate knowledge representation model for bridging the gap between the human requirements and the computational requirements [19]–[22], thus able to play the role of both Computation-Independent Model (CIM) and Platform-Independent Model (PIM). Following MDA, the ontology-based choreography representations are successively transformed through a set of processes until the platform-specific choreography (O_{PdDd}) that is intended to be executed in a specific VWP. The proposed architecture is depicted in **Fig. 3**, and comprehends four models (ontologies) and five processes.

3.1 Choreography Models

We use four different models to represent different abstractions of the choreography that are represented by the following ontologies:

- O_{PiDi} is the generic high-level ontology representing the core concepts of a choreography independent of any implementation platform, also designated as the foundational choreography [23]. **Fig. 2.** depicts its current status.

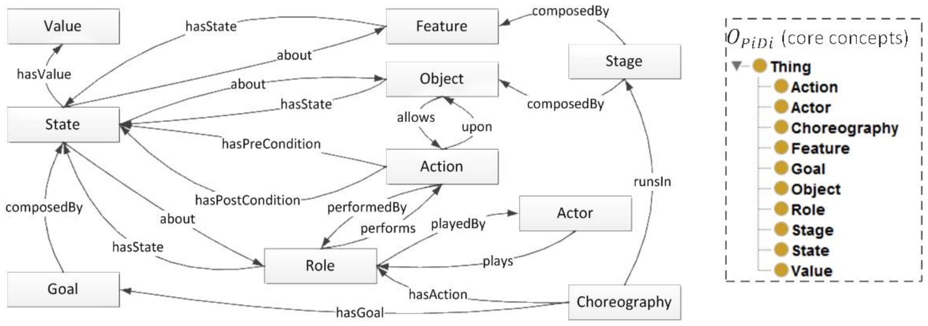


Fig. 2. The Foundational Choreography Ontology (O_{PiDi}): representation of concepts, properties and relations between concepts.

- O_{PdDi} represents the core concepts of a choreography for a specific VWP. It is developed only once, but may have to be adapted in the future to reflect the evolution of the VWP. This ontology should capture the interpretation of a choreography for a VWP in a private way in order to best fit its characteristics, but must capture every concept corresponding to those defined in the foundational ontology to represent the same semantic knowledge, so that is possible to establish semantic relations between them.
- O_{PiDd} is a choreography resulting from the authorship of O_{PiDi}. It captures the representation of a complete choreography for a specific application domain, without any commitment to the technical specificities of any platform.
- O_{PdDd} is the representation of a choreography for a specific VWP.

3.2 Processes

The adaptation of a generic choreography to a specific choreography of a particular VWP is conducted through a set of five processes that execute successive transformations of the models representing different abstractions of choreography.

Authoring. It is a user-based process in which the choreographer authors a domain dependent choreography in the form of an ontology.

The foundation ontology (O_{PiDi}) is extended and refined semantically to describe the choreography entities specific to an application domain, giving rise to O_{PiDd}, but the modifications applied cannot change the semantics defined in O_{PiDi}. For that, the following assumptions must be guaranteed:

- No axioms defined in O_{PiDi} can be removed;
- No contradictions can be added, i.e., O_{PiDd} must be logically consistent;
- No new root elements are allowed. I.e. new entities (classes and properties) are defined as sub entities of those existing in O_{PiDi} and should not create new root elements.

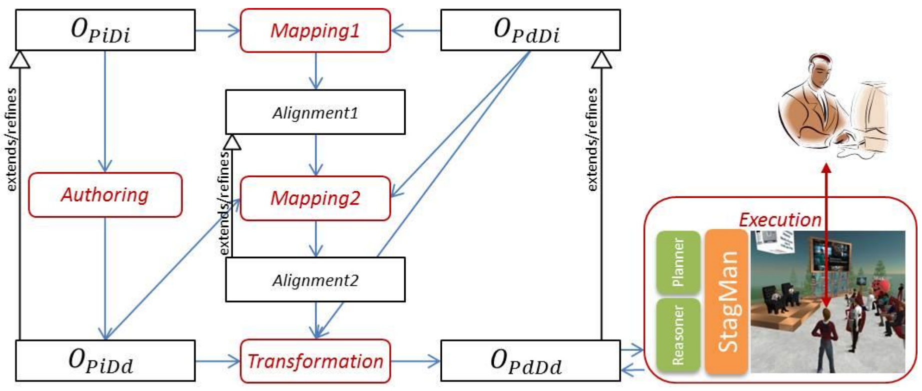


Fig. 3. - The system architecture with the processes of authoring, mappings, transformation and execution

In this process, new concepts are added as well as restrictions to define the type of action actors can perform. Restrictions are associations between roles (that actors can play) and actions, and has a dual purpose: (1) to allow an actor performing an action if he plays a role compatible with it, and (2) to assign dynamically a role to an actor according to the type of action he performs.

For a better understanding, an illustrative example for these two possible constraints is presented.

1. Assuming that the type of action Drop_SDM1 can only be performed by an actor who plays the role T1, the specification of the constraint using the Manchester syntax can be as the following:

```
Class: Drop_SDM1
    SubClassOf:
        upon some SDM1,
        hasPostCondition some s13,
        hasPreCondition some s12,
        performedBy some T1,
        performedBy only T1
```

2. It is intended that, when a Technician performs any of the actions: Drop_SDM1, Go_P1, Grab_SDM1 or Lift_Screw1, his/her role is automatically changed to role

T1. The rule that specifies this assignment can be defined using the Manchester syntax as follows:

```
Class: T1
    EquivalentTo:
        Technician
         and ((performs some Drop_SDM1)
         or (performs some Go_P1)
         or (performs some Grab_SDM1)
         or (performs some Lift_Screw1))
    SubClassOf:
        Technician
```

Mapping1. It is the process that establishes correspondences between the representations of the two choreographies abstractions represented by $O_{P_i D_i}$ and $O_{P_d D_i}$ (Alignment1).

Mapping2. It is the process that establishes correspondences between the domain choreography (O_{PiDd}) and a VWP ontology (O_{PdDi}), i.e. Alignment2. Alignment2 profits from (and extends) Alignment1, thus promoting reuse and reducing effort.

Transformation. It is the process that creates the VWP choreography (O_{PdDd}) from O_{PiDd} and O_{PdDi} and according to the Alignment2.

This is a fully automatic process that "copies" the O_{PiDd} concepts and constraints to the O_{PdDd}. This process is described in section 4. Despite this being an automatic process, choreographers can edit the resulting O_{PdDd} ontology for additional adjustments to better fit the implementation platform.

Execution. It is the process that stages the choreography in a VWP through a Staging Manager (StagMan in **Fig. 3**) that is a computer program compatible with the VWP. It has the following responsibilities: (1) monitors the VW and the behavior of actors, (2) updates the Knowledge Base (KB) according to the changes to the states of VW and the actions performed by actors, (3) validates the actions triggered by actors comparing them with those described in the choreography, and (4) gives feedback to actors about the actions they triggered through the VW interface.

4 Generating a Choreography

Running the aforementioned transformation process generates a choreography to a specific VWP. This fully automatic process gives rise to the O_{PdDd} by performing a novel ontology transformation algorithm. This process may be defined by the function $f\colon transformation(O_{PiDd}, O_{PdDi}, A_2) \rightarrow O_{PdDd}$ such that A_2 is the alignment resulting from the alignment of the O_{PiDd} and O_{PdDi} ontologies. The transformation process consists of the following steps:

1. The taxonomy of each O_{PiDd} ontology concept (hierarchical structure considering the concept as root) added during the authoring process is replicated into the O_{PdDd} according to the correspondences in A_2. The whole sub-taxonomy of the concept is copied to ensure that all expressiveness and restrictions developed in the authoring process will be available in the target ontology.
2. For each copied sub-concept in the target ontology (O_{PdDd}), their properties and relations will be set based on the alignments. Simultaneously, the restrictions applied to each source sub concept are also replicated in the corresponding target sub concept in accordance to the defined alignments.

The algorithm executed in this process is depicted in **Fig. 4.** Consider CB and PB as correspondences (or semantic relations/bridges). CB (Concept Bridge) represents the correspondence between one concept from the source ontology and one concept from the target ontology. PB (Property Bridge) represents the correspondence between one property from the source concept and one property from the target concept.

```
Transformation(Alignment A2) {
  processConceptBridges(A2)
  processPropertyBridges(A2)
}
//[step 1]
processConceptBridges(Alignment A2) {
  for each CB in A2.getListOfConceptBridges() {
    SC = CB.getSrcConcept()
    TC = CB.getTgtConcept()
    createConceptHierarchy(SC, TC)
  }
}
createConceptHierarchy(SC, TC) {
  for each srcSubConcept in SC.getListOfSubConcepts(){
    if(not A2.existCBforSrcConcept(srcSubConcept)){
      tgtSubConcept = TC.createSubConcept(srcSubConcept)
      createConceptHierarchy(srcSubConcept, tgtSubConcept)
    }
  }
}
//[step 2]
processPropertyBridges(Alignment A2) {
  for each CB in A2.getListOfConceptBridges() {
    SC = CB.getSrcConcept()
    TC = CB.getTgtConcept()
    for each srcSubConcept in SC.getListOfSubConcepts() {
```

```
   if(not A2.existCBforSrcConcept(srcSubConcept)){
    tgtSubConcept = TC.getCrrsSubConcept(srcSubConcept)
    listPB = CB.getListOfPropertyBridges()
    for each PB in listPB) {
     SP = PB.getSrcProperty()
     TP = PB.getTgtProperty()
     TC.createProperty(TP)
    }
    if(srcSubConcept.hasRestrictions()){
     TR = tgtSubConcept.createRestriction()
     for each SR in srcSubConcept.getListOfRstrcts() {
      TR.copyRestriction(SR)
     }
    }
   }
  }
 }
}
```

Fig. 4. The transformation process algorithm

To better understand how the transformation process works, we present a small example. Let us assume that in the authoring process it is intended to represent only two types of action: GrabScrewDriver and SayStart. For the sake of organization, GrabScrewDriver is considered as a subtype of ManipulateAction and SayStart a subtype of commandAction. There are two distinct types of roles: Supervisor and Mechanic which in turn has a subtype M1. The type ManipulateAction must be performed by Mechanic and the type CommandAction by Supervisor.

The choreography will be transformed for a messaging platform (developed in-house) and OpenSim[1] (OpenSim). The messaging platform only models one action (Write) and OpenSim models the types: Touch, Chat, Detach, Attach and Go. The rest of example will address the messaging platform only.

In Alignment1 CBs and PBs are defined to link the elementary concepts and properties of O_{PiDi} and O_{PdDi}, e.g., CB1 links O_{PiDi}:Action with O_{PdDi}:Action, CB2 links O_{PiDi}:Role with O_{PdDi}:Role, and PB1 links O_{PiDi}:performedBy with O_{PdDi}:writtenBy.

In Alignment2 are defined new alignments for the types of action that have been added during the authoring. In the messaging platform there is only one action, hence all new types of action of O_{PiDi} should be aligned with it (simply align the new top concepts). We define two CBs (CB1.1 and CB1.2) to align the two new types of action (O_{PiDi}:ManipulateAction and O_{PiDi}:CommandAction) with O_{PdDi}:Write.

Running the transformation algorithm, and based on the defined alignment (Alignment2), the processing of CBs and PBs creates new concepts and properties in O_{PdDd}, described as follows:

[1] http://opensimulator.org/

1. In the first step are processed the concepts. For each CB, is created in O_{PdDd} ontology the hierarchy of the concept of O_{PiDd} that is aligned. This is done by the method *createConceptHierarchy(SC, TC)* ([step1] lines 6-12 of the algorithm);
2. In the second step the properties are processed. For each CB, all sub-concepts of the O_{PiDd} concept are scanned, and according to all PBs associated with this concept, the corresponding property in O_{PdDd} ([step2] lines 30-34) is created and after that, the existing restrictions are added ([step2] lines 35-40).

Although this is an automatic process, choreographers can intervene and edit the resulting O_{PdDd} ontology for additional adjustments. Thus, the O_{PdDd} ontology may be further tweaked to best suit the implementation platform. **Fig. 5** depicts an extract of a O_{PdDd} ontology for Messaging and OpenSim platforms resulting from the concept-bridge transformation process.

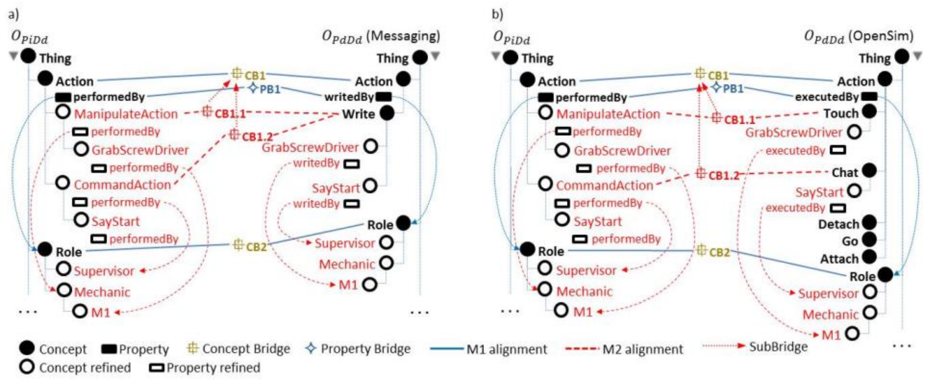


Fig. 5. An extract of O_{PdDd} ontology for a) Messaging and b) OpenSim

Given the representation of a choreography it is important to check the feasibility of its staging. Some factors may lead to the impossibility of its execution, e.g. the virtual world can be in a state incompatible with the choreography or actions having cross dependences between them causing deadlock situations.

The validation of the feasibility of the choreography is performed using a planner that, from the current state of the virtual world and the representation of actions of the choreography (with pre and post conditions) calculates a plan of actions capable of achieving the set of goals. When a solution for the problem is not found, it means that you cannot proceed with the execution of the choreography, or the choreography goals have been achieved in the meantime. The validation of the choreography can be done earlier after the transformation process, and later during its staging when there are suspicions of passivity or deadlocks.

The validation of choreography can be performed at an early stage after the transformation process or at any time during its staging, especially when there are no progresses. In this case the validation can be used to check if there are deadlocks or simply passive actors.

5 Executing a Choreography

Another important key point is to ensure that during the staging of a choreography the ontology remains consistent, i.e., semantically valid. The staging in each of the virtual worlds is controlled by StagMan that continuously monitors the state of the world and the behavior of actors, and when it senses that the behavior of an actor corresponds to a type of action of the choreography, it performs a progressive validation following the next three steps:

1. Checks the KB whether the necessary pre-conditions for this type of action are met;
2. Creates an instance of this action in the KB relating it to the actor that requested it;
3. Apply the post-conditions described by this type of action.

The result of the validation is communicated to the actor responsible for the action through the VWP interface, and is one of the following messages: (1) "Preconditions are not met" when step 1 fails, (2) "Unauthorized Action" when step 2 or step 3 fails, and (3) "Correct" when the three steps are successful.

The consistency checking of the model is automatic and permanent. This procedure is carried out by a reasoning tool, which is coupled to the StagMan component. Any changes made inside the KB, i.e. insertion or removal of facts and changes in relationships automatically trigger the execution of the reasoner that checks if the semantic integrity of the KB remains consistent. When the attempting to execute an action causes inconsistencies in the KB, the action is ignored and the changes associated with it are canceled and the information is restored to the state prior to the action.

6 Experiments

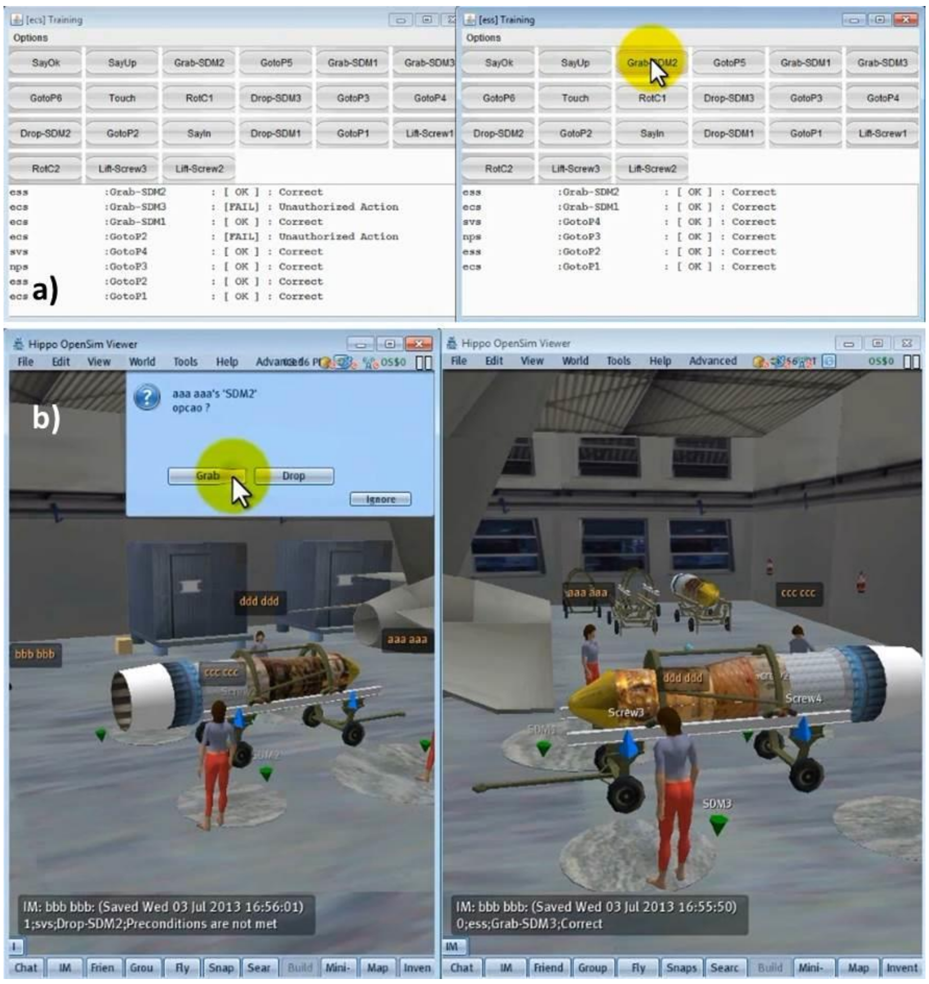
To evaluate our approach we deployed several real-world choreographies that were staged in three different multiuser platforms with very distinct characteristics. We selected examples involving quite different application domains, such as maintenance of military equipment, sports games and children's stories. After authoring the choreographies, the resulting ontologies undergo the remaining processes (Mapping1, Mapping2, Transformation) and staged in different VWP, as depicted in **Fig. 6** and **Fig. 7**.

We used OpenSim, to create a realistic multiuser 3D environment, SmartFoxServer[2] (SmartFox) that allow creating massive multiplayer applications with an isometric environment 2.5D using OpenSpace[3] engine and as a counterpart system, we developed for testing purposes a messaging platform. This messaging platform has characteristics that differ greatly from OpenSim and SmartFox, since it does not allow the graphical representation of scene objects, but enables the development of a team's choreography nonetheless. In the messaging platform interface, the actions activation

[2] http://www.smartfoxserver.com/
[3] http://www.openspace-engine.com/

of the choreography are presented in the form of Buttons, thus the interaction is done by pressing them, and when an action is performed successfully by the team member, it is communicated to all other team members by means of a text log (**Fig. 6** a).

In OpenSim platform, (i) the actions that can be performed on an object are presented in the form of a menu when an actor selects the object, (ii) verbal actions are implemented through the existing Chat mechanism of the virtual world, and (iii) the movement of avatars is known through sensors placed on the stage.



```
ess          :Grab-SDM2    : [ OK ] : Correct
ecs          :Grab-SDM3    : [FAIL] : Unauthorized Action
ecs          :Grab-SDM1    : [ OK ] : Correct
ecs          :GotoP2       : [FAIL] : Unauthorized Action
svs          :GotoP4       : [ OK ] : Correct
nps          :GotoP3       : [ OK ] : Correct
ess          :GotoP2       : [ OK ] : Correct
ecs          :GotoP1       : [ OK ] : Correct
```

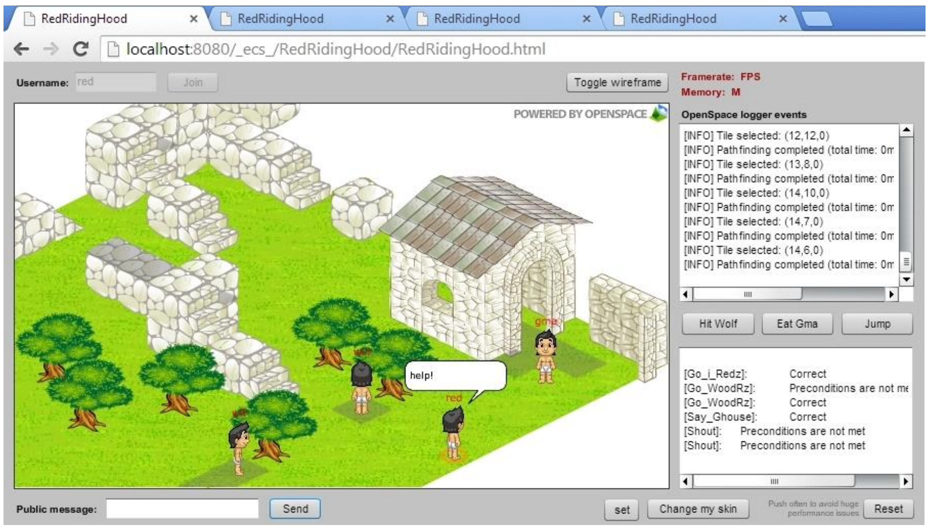
```
ess          :Grab-SDM2    : [ OK ] : Correct
ecs          :Grab-SDM1    : [ OK ] : Correct
svs          :GotoP4       : [ OK ] : Correct
nps          :GotoP3       : [ OK ] : Correct
ess          :GotoP2       : [ OK ] : Correct
ecs          :GotoP1       : [ OK ] : Correct
```

```
IM: bbb bbb: (Saved Wed 03 Jul 2013 16:56:01)
1;svs;Drop-SDM2;Preconditions are not met
```

```
IM: bbb bbb: (Saved Wed 03 Jul 2013 16:55:50)
0;ess;Grab-SDM3;Correct
```

Fig. 6. The staging of the procedures to insert an engine into an aircraft in a) Messaging and b) OpenSim platforms

In SmartFox platform, non-verbal actions are associated with buttons in the interface, verbal actions are interpreted through public messages that actors exchange between them, and actions related to movement of avatars are determined by sensors placed on the stage.



```
[INFO] Tile selected: (12,12,0)
[INFO] Pathfinding completed (total time: 0m
[INFO] Tile selected: (13,8,0)
[INFO] Pathfinding completed (total time: 0m
[INFO] Tile selected: (14,10,0)
[INFO] Pathfinding completed (total time: 0m
[INFO] Tile selected: (14,7,0)
[INFO] Pathfinding completed (total time: 0m
[INFO] Tile selected: (14,6,0)
[INFO] Pathfinding completed (total time: 0m
```

```
[Go_i_Redz]:        Correct
[Go_WoodRz]:        Preconditions are not me
[Go_WoodRz]:        Correct
[Say_Ghouse]:       Correct
[Shout]:     Preconditions are not met
[Shout]:     Preconditions are not met
```

Fig. 7. The staging of the Little Red Riding Hood story in SmartFox platform

Authoring is obviously the most time-consuming and creative process (the expressivity of the ontologies authored during the experiments range between $\mathcal{ALCHI}(\mathcal{D})$ and $\mathcal{SHOIN}(\mathcal{D})$ Description Logics expressivity), the Mapping1 and Mapping2 semi-automatic processes require reduced time and effort. Once these processes are performed, the transformation and execution processes are fully automatic.

7 Conclusions and future work

In this paper we propose an approach that allows the development of choreographies and its adaptation and staging in potentially any VWP. For that, based on the concept of MDA and the assumption that the use of ontologies is possibly the best way to represent the conceptual information to approximate the intelligent systems to the human conceptual level, we propose an ontology to capture the semantics of a generic choreography independent of any application domain and VWP. Further, for each VWP is adopted an ontology representing its specific features that is mapped with the generic one.

Moreover, ontologies allow the integration in the same model of all the modeling information related to the choreography, i.e. the definition of procedures related to teamwork and the information about the scene.

The use of alignments between ontologies enables the automation of the adaptation of the generic ontology to the specific target ontology, hence contributing to reduce development time and resources.

We describe a sequence of processes that uses alignments between ontologies and allows adapting a choreography of a specific domain into a choreography capable of being staged in potentially any VWP. In particular this paper presents the fully automatic algorithm for transforming a platform independent yet domain dependent choreography (i.e. a model) into a platform dependent and domain dependent choreography. The generated choreography is applied by the execution process (a platform dependent process) for staging the choreography and allowing and controlling the user interaction.

In future work the Mapping1 and Mapping2 processes can be refined to incorporate automatic matching mechanisms based on past experiences with the same VWP and choreographies' domains. So, it would be possible to increase the ability to automate these processes while at the same time it reduces the need for user intervention.

8 Acknowledgments

This work is supported by FEDER Funds through the "Programa Operacional Factores de Competitividade - COMPETE" program and by National Funds through FCT "Fundação para a Ciência e Tecnologia" under the project AAL4ALL (QREN13852).

9 References

[1] S. De Freitas, 'Serious virtual worlds', *Scoping Guide JISC E-Learn. Programme Jt. Inf. Syst. Comm. JISC UK*, 2008.

[2] L. Morgado, J. Varajão, D. Coelho, C. Rodrigues, C. Sancin, and V. Castello, 'The attributes and advantages of virtual worlds for real world training', *J. Virtual Worlds Educ.*, vol. 1, no. 1, 2010.

[3] P. Kapahnke, P. Liedtke, S. Nesbigall, S. Warwas, and M. Klusch, 'ISReal: An Open Platform for Semantic-Based 3D Simulations in the 3D Internet', in *The Semantic Web – ISWC 2010*, vol. 6497, P. Patel-Schneider, Y. Pan, P. Hitzler, P. Mika, L. Zhang, J. Pan, I. Horrocks, and B. Glimm, Eds. Springer Berlin / Heidelberg, 2010, pp. 161–176.

[4] A. Pinheiro, P. Fernandes, A. Maia, G. Cruz, D. Pedrosa, B. Fonseca, H. Paredes, P. Martins, L. Morgado, and J. Rafael, 'Development of a Mechanical Maintenance Training Simulator in OpenSimulator for F-16 Aircraft Engines', *Procedia Comput. Sci.*, vol. 15, pp. 248–255, 2012.

[5] 'Media Grid : Open File Formats Technology Working Group (OFF.TWG) Charter'. [Online]. Available: http://mediagrid.org/groups/technology/OFF.TWG/. [Accessed: 14-Oct-2013].

[6] N. Mollet and B. Arnaldi, 'Storytelling in Virtual Reality for Training', in *Technologies for E-Learning and Digital Entertainment*, vol. 3942, Z. Pan, R. Aylett, H. Diener, X. Jin, S. Göbel, and L. Li, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 334–347.

[7] S. Gerbaud, N. Mollet, F. Ganier, B. Arnaldi, and J. Tisseau, 'GVT: a platform to create virtual environments for procedural training', in *IEEE Virtual Reality Conference, 2008. VR '08*, 2008, pp. 225–232.

[8] T. M. Vernieri, ‘A web services approach to generating and using plans in configurable execution environments’, 2006.

[9] S. Alhir, ‘Methods & Tools - Understanding the Model Driven Architecture (MDA)’, Martinig & Associates, fall 2003.

[10] L. Edward, D. Lourdeaux, D. Lenne, J. Barthes, and J. M. Burkhardt, ‘Modelling autonomous virtual agent behaviours in a virtual environment for risk’, *IJVR Int. J. Virtual Real.*, vol. 7, no. 3, pp. 13–22, 2008.

[11] A. Lopes, B. Pires, M. Cardoso, A. Santos, F. Peixinho, P. Sequeira, and L. Morgado, *System for Defining and Reproducing Handball Strategies in Second Life On-Demand for Handball Coaches ’ Education*. .

[12] R. M. Young, M. O. Riedl, M. Branly, A. Jhala, R. J. Martin, and C. J. Saretto, ‘An architecture for integrating plan-based behavior generation with interactive game environments’, *J. Game Dev.*, vol. 1, no. 1, pp. 1–29, 2004.

[13] R. M. Young, J. Thomas, C. Bevan, and B. A. Cassell, ‘Zócalo: A Service-Oriented Architecture Facilitating Sharing of Computational Resources in Interactive Narrative Research’, 2011.

[14] S. P. Cash and R. M. Young, ‘Bowyer: A Planning Tool for Bridging the gap between Declarative and Procedural Domains’, *Artif. Intell.*, pp. 14–19, 2009.

[15] ‘MBUI - Task Models’. [Online]. Available: http://www.w3.org/TR/task-models/. [Accessed: 09-Jul-2014].

[16] R. E. Fikes and N. J. Nilsson, ‘STRIPS: a new approach to the application of theorem proving to problem solving’, in *Proceedings of the 2nd international joint conference on Artificial intelligence*, San Francisco, CA, USA, 1971, pp. 608–620.

[17] M. Lekavỳ and P. Návrat, ‘Expressivity of STRIPS-like and HTN-like planning’, *Agent Multi-Agent Syst. Technol. Appl.*, pp. 121–130, 2007.

[18] E. Silva, N. Silva, and L. Morgado, ‘Staging Choreographies for Team Training in Multiple Virtual Worlds Based on Ontologies and Alignments’, presented at the HCI International 2014, Heraklion, Crete, Greece (accepted for publication), 2014.

[19] L. Obrst, H. Liu, and R. Wray, ‘Ontologies for Corporate Web Applications’, *AI Mag.*, vol. 24, no. 3, p. 49, Sep. 2003.

[20] T. R. Gruber and others, ‘A translation approach to portable ontology specifications’, *Knowl. Acquis.*, vol. 5, no. 2, pp. 199–220, 1993.

[21] D. Fensel, *Ontologies: a silver bullet for knowledge management and electronic commerce*. Springer, 2004.

[22] M. Gruninger and J. Lee, ‘Ontology Applications and Design-Introduction’, *Commun. ACM*, vol. 45, no. 2, pp. 39–41, Feb. 2002.

[23] E. Silva, N. Silva, H. Paredes, P. Martins, B. Fonseca, and L. Morgado, ‘Development of platform-independent multi-user choreographies for virtual worlds based on ontology combination and mapping’, in *2012 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, 2012, pp. 149 –152.