



GRAPPLE

D4.1a Version: 1.0

Design of the Infrastructure for the Adaptive Delivery of Simulations

Document Type	Deliverable
Editor(s):	Martin Harrigan (TCD), Vincent Wade (TCD), Ian O’Keeffe (TCD)
Author(s):	Martin Harrigan (TCD), Vincent Wade (TCD), Ian O’Keeffe (TCD)
Internal Reviewers:	Kees van der Sluijs (TUE), Dimitri Rambout (UCL)
Work Package:	WP4
Due Date:	31-01-09
Version:	1.0
Version Date:	31-01-09
Total number of pages:	40

Abstract: This deliverable details the design of an infrastructure for the delivery of adaptive simulations, in particular, procedural simulations through dialogs and the adaptive combination of dialogs. It provides some motivation for adaptive simulations – why adaptive hypermedia based on content alone is not enough. We survey existing engines for delivering adaptive simulations (or some element of them). We specify the requirements for the delivery of adaptive simulations. We also present an initial design for such an infrastructure.

Keyword list: adaptive simulations, delivery, infrastructure, tool set

Summary

Deliverable D4.1a (Design of the Infrastructure for the Adaptive Delivery of Simulations) has been produced for the EU FP7 STREP project, GRAPPLE (Generic Responsive Adaptive Personalized Learning Environment). The GRAPPLE project aims at delivering to learners a technology-enhanced learning (TEL) environment that guides them through a learning experience, automatically adapting to personal preferences, prior knowledge, skills and competences, learning goals and the personal or social context in which the learning takes place. The same TEL environment can be used at home, school, and work or on the move (using mobile/handheld devices). The authoring and delivery of adaptive simulations will be the focus of Deliverables D3.5a, D3.5b, D3.5c, D4.1a, and D4.1b. The scope of this deliverable is to detail the design of an adaptive engine for simulations, in particular, procedural simulations through dialogs and the adaptive combination of dialogs.

The purpose of this deliverable is:

- To provide some motivation for adaptive simulations – why adaptive hypermedia based on content alone is not enough;
- To survey existing engines for delivering adaptive simulations (or some element of them);
- To specify the requirements for the delivery of adaptive simulations; and
- To present an initial design for such an infrastructure.

The main contributions of this deliverable are:

- An extended set of requirements for the delivery of adaptive simulations; and
- An initial design based on workflow engines and Portal technology that will be used for the first implementation of this infrastructure.

Authors

Person	Email	Partner code
Martin Harrigan	martin.harrigan@cs.tcd.ie	TCD
Vincent Wade	vincent.wade@cs.tcd.ie	TCD
Ian O’Keeffe	okeeffei@cs.tcd.ie	TCD

Table of Contents

1	INTRODUCTION	6
2	RELATED WORK	7
3	REQUIREMENTS	9
4	ARCHITECTURE	11
4.1	The Models	13
4.2	The Workflow Engine	14
4.2.1	Starting a Workflow	15
4.2.2	Parallel Branching	15
4.2.3	Workflow State.....	15
4.2.4	Interruptible Services.....	15
4.3	The Adaptive Engine	16
4.4	The Portal	17
4.5	Author-Time, Publish-Time, and Run-Time	18
4.6	The GRAPPLE Adaptation Language	18
5	CONCLUSIONS	21
A	DEFINITIONS	26
5.1	Resource Model	26
5.2	Domain Model	26
5.3	Pedagogical Relationship Types	27
5.4	Adaptation Model	27
5.5	Activities	27
B	COMMON XML SCHEMA	28
C	XML SCHEMA FOR THE DOMAIN MODEL	29
D	XML SCHEMA FOR THE PEDAGOGICAL RELATIONSHIP TYPE	31
E	XML SCHEMA FOR THE ADAPTATION MODEL	34
F	EXAMPLE XML FILE FOR THE DOMAIN MODEL	35
G	EXAMPLE XML FILE FOR THE PEDAGOGICAL RELATIONSHIP TYPE	38

H EXAMPLE XML FILE FOR THE ADAPTATION MODEL 39

Tables and Figures

List of Figures

Figure 1: An abstract workflow for a medical application..... 7
 Figure 2: A portal with integrated workflow displaying a list of pending workflow tasks. 8
 Figure 3: Our architecture for delivering adaptive simulations..... 12
 Figure 4: Our architecture for delivering adaptive simulations is based on Portal technology. The grey set of rectangles is brokers for the individual WSRP interfaces. 12
 Figure 5:GALE's architecture for the GRAPPLE project. (Reproduced from deliverable D1.3a.) ... 17
 Figure 6: An aggregation of portlets for inspecting financial data (©2009 Google: The iGoogle interface). 18

List of Acronyms and Abbreviations

AM	Adaptation Model
APeLS	Adaptive Personalized eLearning Service
AWES	Adaptive Web-Based Educational Systems
AWSC	Adaptive Web Service Composition
API	Application Programming Interface
BPML	Business Process Modelling Language
CDM	Concept Domain Model
CRT	Concept Relationship Type
CSRT	Concept-Service Relationship Type
CAM	Conceptual Adaptation Model
DOM	Document Object Model
DM	Domain Model
GRAPPLE	Generic Responsive Adaptive Personalized Learning Environment
GAL	GRAPPLE Adaptation Language
GALE	GRAPPLE Adaptation Engine
ITS	Intelligent Tutoring Systems
IEEE LOM	IEEE Learning Object Metadata
IMS VDEX	IMS Vocabulary Definition Exchange
LMS	Learning Management Systems
PRT	Pedagogical Relationship Type
RM	Resource Model
RDF	Resource Description Framework
SDM	Service Domain Model

SRT	Service Relationship Type
SWM	Service Workflow Model
SCORM	Sharable Content Object Reference Model
SPARQL	SPARQL Protocol and RDF Query Language ¹
TEL	Technology-Enhanced Learning
UM	User Model
WSBPEL	Web Services Business Process Execution Language
WSDL	Web Services Description Language
WSDM	Web Services Distributed Management
WSRF	Web Services Resource Framework
WSRP	Web Services for Remote Portlets
XML	Extensible Markup Language

¹ This is a “recursive acronym”.

1 Introduction

Delivering static content and delivering dynamic content are very different tasks. Hypermedia and adaptive hypermedia are predominantly concerned with static content. The user reads or looks at material and resources and moves on. Interactivity is limited to simple forms or a choice of links. On the web, the first foray from static HTML pages into real interactivity involved Java applets. Similarly, adaptive hypermedia's first experiments with activity involved 'intelligent content' [10][11][15]. However, embedding pockets of isolated intelligence in a sea of content is insufficient. The intelligence is not re-usable and interoperability is non-existent. We require the intelligence to be omnipresent and to extend across 'pages' and users.

Within the GRAPPLE project, we aim to extend the scope of adaptive learning environments to cater for adaptive simulations. Simulations are inherently interactive and activity-based. The authoring of adaptive simulations is a difficult task and is dealt with in deliverable D3.5a. However, the delivery of adaptive simulations is equally difficult and requires functionality that is not present in current adaptive engines. In this deliverable we integrate three distinct technologies to provide an infrastructure for adaptive simulations: adaptive engines, workflow engines, and portals. We show how this combination can provide adaptive selection, sequencing and presentation of both content and services (elemental pieces of an activity) in a unified manner. This approach mirrors some existing non-adaptive systems [7].

There are many advantages in using simulations in an educational context. Most importantly, the learning content can be provided at the opportune time (i.e. mid-activity) and the 'learning transfer' phase is short. Many in the e-Learning community have recently heralded its advent:

"Researchers will continue to make progress in discovering evidence-based principles for the design of e-learning, including new applications of the science of learning to educational games, simulations, and pedagogical agents." – Richard E. Mayer, University of California, Santa Barbara, USA, in Predictions for 2009, eLearn Magazine (<http://www.elearnmag.org>), ACM, Inc.

Furthermore, there are unique advantages in using *adaptive* simulations in an educational context:

- Adaptive simulations become more interesting and engaging. Adaptation occurs at the individual level rather than simply 'classing' learners, as do many simulations, e.g. novice, intermediate, or expert.
- They reduce the burden of composition. The author does not need to develop many similar simulations for variations of the same process or for slightly different audiences.
- There is a clean separation of concerns: the simulation procedural model, the adaptation model and the content model. Most procedural simulations tend to combine the procedural simulation model and the content model, thereby making it difficult to introduce adaptivity.

Delivering adaptive content is a subset of delivering adaptive simulations. Our infrastructure can subsume the more traditional "adaptive engine only" solutions. At its simplest level, an adaptive simulation can just provide learning content in a predefined sequence in much the same as an adaptive hypermedia system does. However, the real benefit only becomes apparent when the sequence or process itself is of interest. This can be categorized more specifically as a procedural simulation and it is this type of simulation that we are focusing on.

Our decision to deliver adaptive simulations through a portal (see Section 4.4) provides a standardized approach when integrating this work with Learning Management Systems (LMSs) – a goal of the GRAPPLE project (see Work Package 7). A portal aggregates portlets or learning services and activities from many different sources into a single coherent interface or point of access. A portlet is a software component with a user interface that is contained in a portlet container and displayed (and possibly aggregated with other portlets) by a portal. There is much overlap between the functionality of an LMS and a portal. LMSs provide portal-type functionality and many, for example uPortal [23], can natively host portlets that conform to JSR-168 [3] and JSR-286 [4]. Combining an adaptive engine, a workflow engine, and a portal so that they can work in unison to deliver adaptive simulations is the main focus of this deliverable.

This deliverable is organized as follows. In Section 2 we review some related work from the fields of adaptive hypermedia and adaptive web service composition. We extend the requirements of the GRAPPLE infrastructure for adaptive non-simulated learning environments to cater for adaptive simulations in Section 3. In Section 4 we describe in detail the architecture of our solution. In particular we present the roles of the data models, the workflow engine, the adaptive engine, and the portal. We also describe how the GRAPPLE Adaptation Language (GAL) will cater for adaptive simulations. Finally, we conclude in Section 5.

2 Related Work

Developing an infrastructure for the delivery of adaptive simulations is a novel task. However, there are existing systems that in some way approach this task.

O’Keeffe et al. [34] use Adaptive Web Service Composition (AWSC) to adaptively sequence activities for the Adaptive Personalized eLearning Service (APeLS). They promote a service-oriented approach to personalized learning activities. They point out the similarities and differences between AWSC and adaptive hypermedia (summarized in Section 3) and show how the former can be used to compose adaptive service-oriented applications. The adaptive engine from this work was used in the EU FP6 IST project iClass to provide rich personalized learning experiences to K12 school children.

Ardissono et al. [6] present a framework based on a hierarchical workflow representation supporting an extensible specification of context-sensitive workflows, which can be executed by standard workflow engines (see Figure 1). In their approach, the entire workflow definition is expanded at the beginning to cover all possible adaptations and then left to the workflow engine for execution. In contrast, our architecture will adopt a piecemeal approach when controlling the workflow engine, sending the workflow definition in parts as they become needed.

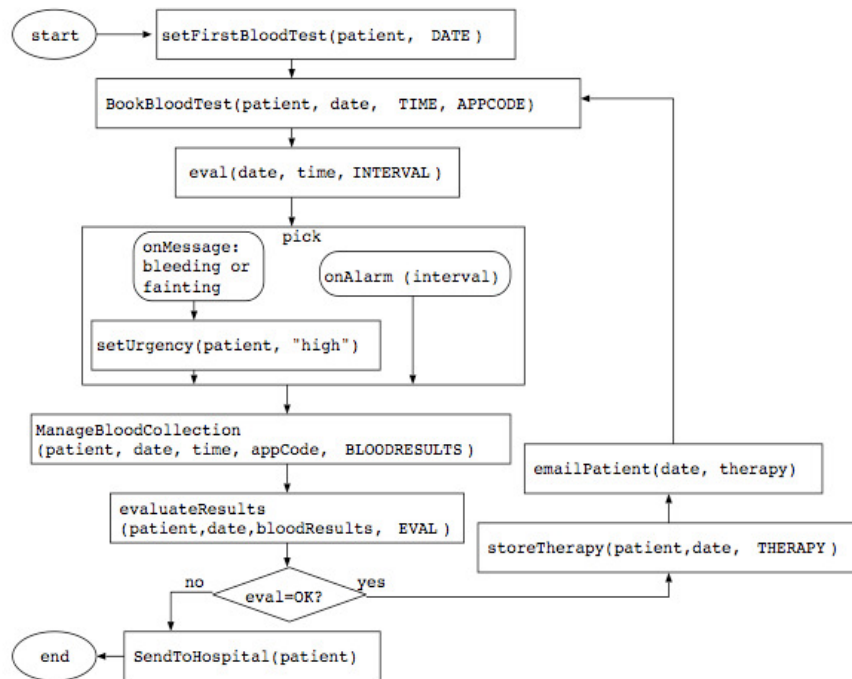


Figure 1: An abstract workflow for a medical application.

De Jong et al. [18] consider the integration of simulation and learning support when studying classical mechanics, in particular, collisions between moving bodies. They experiment with models of varying fidelity and differing levels of intrusiveness by the learning support system.

Adaptive Web-Based Educational Systems (AWESs) are a recognized class of adaptive Web systems [13]. Adaptive textbooks [16][19][48], adaptive quizzes [42], intelligent solution analyzers [47], adaptive class monitoring systems [33] and adaptive collaboration support systems [43] all show considerable benefits over their non-adaptive counterparts. Adaptive Learning Systems (ALSs) should be no different. However, due mainly to their architecture, they have not been widely adopted. They integrate poorly, and do not support reuse. KnowledgeTree [10][11][15] aims to fix this. KnowledgeTree is an architecture for developing component-based ALSs using distributed reusable intelligent learning activities. It is a bridge between popular Web-based Learning Management Systems (LMSs) [14], Intelligent Tutoring Systems (ITSs), and adaptive hypermedia.

KnowledgeTree assumes the presence of at least four kinds of servers or services: activity servers (hosts reusable learning content, content can be 'active' and possibly intelligent and adaptive by communicating with the student model servers); value-adding services (located between a portal and an activity server, it can add adaptive sequencing, annotation, visualization, or content integration); learning portals (provides a single login-point for students and structures access to various distributed fragments, like in an LMS); and student model servers (stores the needs and the prospects of students in the process of e-Learning). These servers or services represent the interests of the three main stakeholders in the e-Learning process: the content and service providers, the course providers, and the students.

KnowledgeTree includes two types of portals – the KnowledgeTree portal which provides primary and secondary learning material and the KnowledgeSea portal which provides a matrix-based knowledge map. The KnowledgeSea portal also provides a form of adaptive navigation support known as social navigation [20]. Value-adding activity services [9][36] include NavEx, which adaptively restricts access to programming examples that the student is ready to learn through visual cues like green and red bullet points (domain dependent), and AnnotatED, which provides the ability to annotate static and interactive content (domain independent). There is some separation between the adaptation and the interface through which the activities are presented to the learner. However, in our architecture this separation will be much more explicit, i.e. the adaptive engine and the portal.

The common protocols include a transparent login protocol or single sign-on (KnowledgeTree uses a simple session ID), a resource discovery and delivery protocol, e.g. a centralized broker-based approach or peer-to-peer [30], and a student modelling protocol. KnowledgeTree uses the CUMULATE student-modelling server. It is an event-based centralized student modelling architecture [11]. External and internal inference agents process the flow of events in different ways and update the values in the inference model in the form of pairs (property-value) (e.g. motivation level) and triples (property-object-value). These pairs and triples form the upper level of the model, which can then be queried by activity servers and portals using a standard querying protocol. Properties can be student or group-wise properties. Queries can be value queries or evidence queries, e.g. what events did the agent use to calculate a given value? GRAPPLE's user model and its interface are detailed in D2.1.



Figure 2: A portal with integrated workflow displaying a list of pending workflow tasks.

The integration of portals in an educational environment with workflow engines has been explored previously (see Figure 2) in the aptly named article "Combating Stovepipes: Implementing Workflow in uPortal" [7].

However, this solution does not explicitly cater for adaptivity. Developing distributed adaptive and intelligent educational systems based on shared educational resources have also been explored in the field of ITS [15].

3 Requirements

An important requirement for our system, over and above the requirements for an adaptive non-simulated learning environment, is the provision of both content and services [34][49]. Simply embedding a service into a piece of content and treating it like all other content is a very simplistic approach that ignores the inherent differences between services and content. For example, when services are treated as content, the parameters of each service must be either ignored or set manually and there is no way to provide for the flow of information between services. For our purposes, we consider a service to have inputs, outputs, preconditions and effects. However, a more precise definition, based on Portlet technology, can be found in D3.5a and in Section 4.4.

By utilizing web service technology and the associated service composition approaches, it is possible to adaptively select and aggregate services using manual design, situation calculus, planning graphs or hierarchical task networks [38] in a manner that complements the goals of adaptive hypermedia. Adaptive hypermedia is primarily concerned with the adaptive selection, sequencing and presentation of content to the user [9][12]. The goal is to provide the user with appropriate material from a large hyperspace while keeping the associated benefits of hypermedia. We need to extend these concerns to the adaptive selection, sequencing and presentation of services to the user. Our approach is based on AWSC. This can be divided into two strands:

- 1) Workflow-Based: This involves the manual composition of web services using specialized languages, e.g. WSBPEL [31]; and
- 2) AI Planning Techniques: This involves the automatic composition of services based on their meta-descriptions and functional properties as well as the initial state of the 'world' and the desired goal state [26][27].

We will focus on the first of these two strands. It will be up to the author to completely specify the workflow when designing an adaptive simulation.

AH and AWSC have many similarities. They both combine content or services into a plan to achieve some goal or state. The adaptive axes (what we can adapt to) of AH are similar to the functional and non-functional properties of services. They both use semantic models to describe the 'elemental' resources that are available for composition. They both use sequencing logic that is 'informed' by external information (e.g. user or context models) to compose or choreograph the elements.

However, there are also some important differences. In general, services are parameterized, content is not. Through parameterization, it is possible to adapt the behaviour and functionality of a service. In order to take advantage of this added flexibility, it is important that the mechanism used to select services is aware of the parameterization of the available services and is capable of automatically configuring some of these parameters so as to modify the behaviour of the service to better suit the needs of the user. Complex services can be composed using many individual services. It is necessary to capture the flow of information between these services. Additionally, services can throw errors or exceptions.

Our infrastructure will need to support all of the above properties of services and the orchestration of the services themselves. In particular, it will need to provide:

- Models: The infrastructure will need to provide a common method of access to the various models that an adaptive simulation relies on. For example, all resources (pieces of content or service instances) are delivered to the learners through this infrastructure. Whether a resource originates on the Web or from some other, say, institutional repository is irrelevant to the learner. It may also be the case that an author would like to include his own resources, to which only he and his adaptive simulations can access. The infrastructure will hide this distinction by providing a common interface.
- A workflow engine: We need a workflow engine that can enact the workflow within an adaptive simulation. However, we must be able to change this workflow at the adaptive simulation proceeds. The adaptation is performed 'on the fly' so there is no way of knowing at the beginning, what the entire workflow definition will be. The workflow engine will also need to handle the flow of information between services and be able to persist sessions if the user chooses to pause an adaptive simulation and continue its execution at a later time. To some extent, functionality will be shared between the workflow engine and the adaptive engine (see below) as both will have different data that needs to be saved.

- A portal: We need a unified interface through which we can deliver an adaptive simulation. This will need to be able to aggregate together a variety of content and services so that they appear to be part of 'one' application. In order to allow the learner to interact with multiple services simultaneously, the portal must also support a range of actions more diverse than simply clicking on a link. The user may choose to minimize or maximize a service, close a service, or wait for a time-out.
- An adaptive engine: A critical component of our infrastructure is a GAL-compliant (see Section 4.6) adaptive engine. It will require the ability to express and enact the adaptive selection of both content and services, the adaptive sequencing of those services and adaptive presentation. The adaptive engine will communicate with both the workflow engine and the portal. It will take an adaptive simulation, specified in GAL, provide a piece-meal workflow definition to the workflow engine and access the all-necessary models for the portal.

We can also enumerate what is required in order to adequately express adaptive simulations and their behaviour. It will need to provide for simulation-specific adaptive features (what we can adapt) and simulation-specific adaptive dimensions (what we can adapt to) [9][12]. A list of the former includes:

- Adaptive Content (including Services)
 - *Service Selection*: This is the analogue of adaptive content selection in traditional adaptive hypermedia. Many service instances, modelled in the RM, can fulfil the requirements of a service in the Service Domain Model (SDM)². The final choice may depend on attributes and preferences of the learner. Service selection also includes the appropriate parameterization of a service to make it more intuitive to the learner.
 - *Fidelity*: This describes the rigorousness of the process models and the level of complexity of the simulation. It is the accuracy of the representation when compared to the real world. A simulation is often categorized as having low, medium, or high fidelity. A common mistake is to assume that it is always better to use high fidelity. Low fidelity can be more appropriate at the beginning of a training or learning session. It can be increased as the learner becomes more familiar with the simulation. This is also known as model progression [18][46]. Fidelity can be further divided into constituent factors, for example, accuracy, capacity, error, fitness, precision, resolution, sensitivity, tolerance, validity, etc. This can be considered 'sub-features' of the fidelity feature.
 - *Modality*: A modality is either a sense through which a human can receive output from a computer or a device through which a computer can receive input from a human. In other words, it is a path of communication between a human and a computer. Examples of the former include sight, hearing, touch, balance, temperature, etc. and examples of the latter include display, microphone, speaker, accelerometer, etc. The range of devices attached to a computer limit the modalities that can be offered by a simulation. For example, if the user wishes to use a chat room service they can speak aloud if the computer has a microphone and speaker and the service supports audio transmission, otherwise they may have to type what they wish to say. To improve the educational prospects of a simulation and engage the learners, it is necessary to offer multiple modalities at low, medium and high fidelity levels.
 - *Feedback*: This can be categorized along two dimensions. Firstly, feedback can be either natural or artificial. Natural feedback blends in with the simulation. It mirrors the feedback provided by the real world process. For example, suppose the learner is presented with a service that offers the functionality of a programming language interpreter. This enables the learner to experiment with code for himself. If the interpreter reports a syntax error in the same way as a real world interpreter, then this is a form of natural feedback. On the other hand, suppose the service provides advice for coding style and indentation while the learner is typing in their code, then this is a form of artificial feedback. A real world interpreter does not normally provide this. Secondly, feedback can be either immediate or delayed. It can be provided at the opportune time or delayed until the learner has completed some task.
 - *Motivators*: Motivation, in the context of educational simulations, is a desire to learn. Simulations can include artefacts that explicitly try to increase motivation, e.g. motivational quotes, questions that instil curiosity, competitions between learners, scoreboards, etc. These can be included or omitted depending on the perceived motivation of the learner.

² This is the 'service analog' to the Concept Domain Model (CDM) and is explained in more detail in Annex A.

- Adaptive Navigation / Workflow
 - *Sequencing*: The actual sequencing or workflow of an adaptive simulation can be adapted to the learner. This adaptation can depend on, for example, the role of the user, the number of times they have iterated through a section of the simulation, or the amount of time they can devote to the learning activity. This can be specified through workflow constructs that are augmented with adaptive behaviour.
 - *Control*: Both the system and the learner determine, to a certain extent, the path taken through an adaptive simulation. Control refers to the degree to which either the system or the learner decides this path. At an early stage in the training or learning session, it may be desirable to allow the system to make most of the decisions, guiding the learner through the process. Eventually, it may be entirely the learner's choice as to which path he wishes to follow.
 - *Mapping*: In traditional adaptive hypermedia, a map provides a global overview of the user's position in the hyperspace. It helps prevent the 'lost-in-hyperspace' problem. This is equally important in adaptive simulations. A map can provide the user's position in the overall workflow. It helps show what services the user has finished with, what services are pending, and whether the user is in two or more parallel branches of workflow at the same time. At its simplest level, it can provide a list of services or tasks that the user currently needs to complete (see Figure 2).
- Adaptive Presentation
 - *Arrangement*: Considering services to be portlets that are delivered to the learner through a portal affords new possibilities in adaptive presentation that are not present in traditional adaptive hypermedia. The portlets can be arranged within the portal to suit the user. For example, if two services are required to advance to the next stage of the workflow, then the two corresponding portlets can be displayed side-by-side in the portal. This arrangement can be performed automatically by the system when needed. In traditional adaptive hypermedia, the content is ultimately displayed through a browser. If a learner wishes to open two pieces of content simultaneously he or she needs to open two browsers (or two tabs within the same browser). When presenting adaptive simulations, we need the possibility of having more than one service opened at a time without resorting to browser-based functionality.

A list of the latter, simulation-specific adaptive dimensions, is no more extensive than the list of adaptive dimensions in traditional adaptive hypermedia, e.g. learner goals, competence, language, etc. Deliverables D9.1 and D10.1 have compiled a comprehensive list of these adaptive dimensions and gauged their relative importance as perceived by learners and teachers.

4 Architecture

The main components of our infrastructure are the models (see Section 4.1), a workflow engine (see Section 4.2), an adaptive engine (see Section 4.3), and a portal (see Section 4.4). This provides a separation between execution (a workflow engine), adaptation (an adaptive engine) and presentation or delivery (a portal). These components are described in detail in the following sections.

The architecture, at a very high-level, is presented in Figure 3. The actor in this case is a user or learner, not an author. The tool set and architecture for the authoring task is presented in deliverable D3.5a. The user interacts only with a portal, which can be integrated with a LMS. This portal contains a special container portlet or *outer portlet* for each adaptive simulation. The outer portlet is mapped to an instance of a workflow in the workflow engine. The outer portlet aggregates the inner portlets or services into one 'unified' simulation. An adaptive simulation comprises a collection of inner portlets under the orchestration of a workflow language. These inner portlets could be, for example, chat portlets, discussion portlets, or interpreters for snippets of a programming language. In the context of the GRAPPLE project, the portal will be integrated with existing LMSs and the adaptive engine will be GALE (see deliverable D1.3a).

A complete description of an adaptive simulation is provided to the adaptive engine (see the left-hand side of Figure 3). This is the output of the authoring task. The adaptive engine enacts this output. Portions of GAL code are mapped to an appropriate workflow definition language, e.g. WSBPEL. All access to the models,

e.g. the RM, DM, UM, etc.³, is through the adaptive engine. User actions in the portal can fire events, specified in GAL code, that are handled by the adaptive engine. For example, when the user closes a service the adaptive engine becomes aware of this, and can update the user model in response.

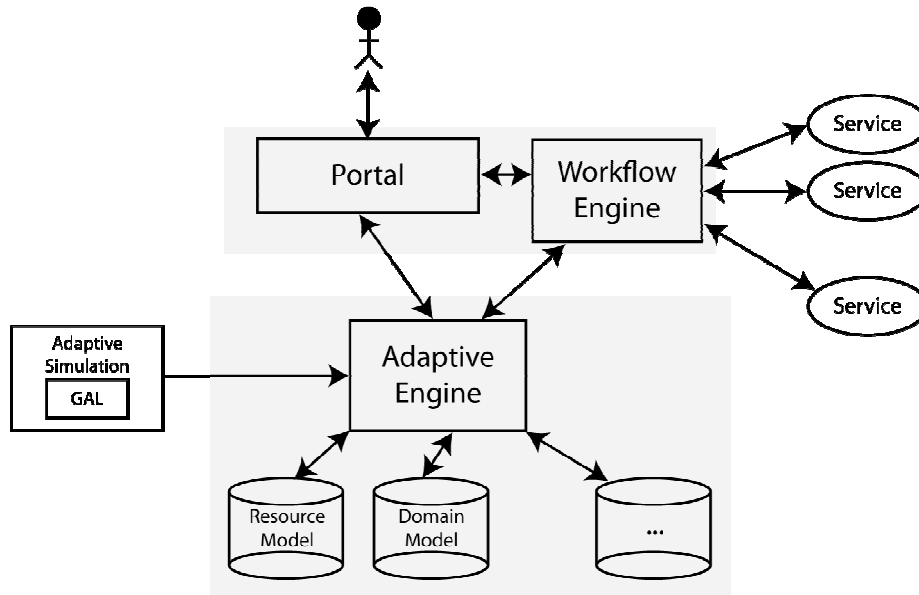


Figure 3: Our architecture for delivering adaptive simulations.

Services are implemented through portlets using the Web Services for Remote Portlets (WSRP) Specification [32]. This allows portlets to be provided through Web Services, and, more importantly, it allows us to broker all interaction with them. The workflow engine is also accessed through a WSRP interface, in the same way as the services that it orchestrates (see Figure 4). This is the interface to the outer portlet described above.

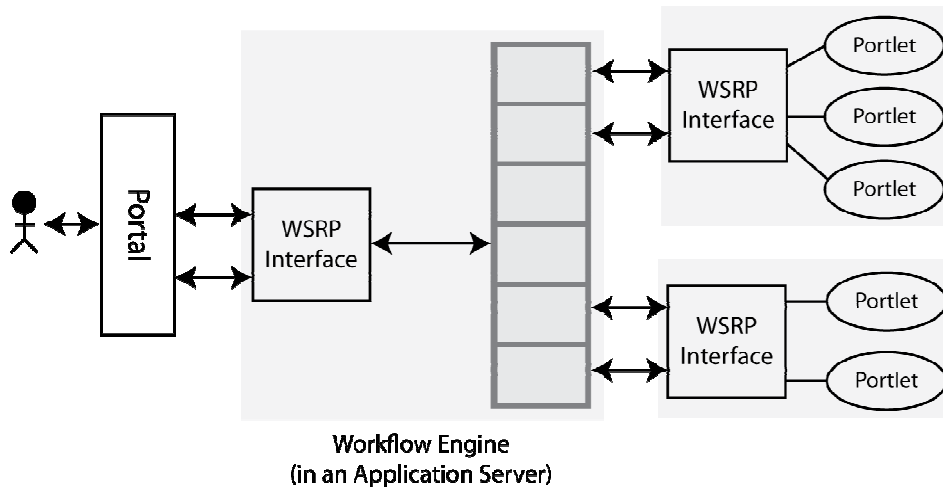


Figure 4: Our architecture for delivering adaptive simulations is based on Portal technology. The grey set of rectangles is brokers for the individual WSRP interfaces.

³ See Annex A for a brief description of each of these models.

When accessed through the portal, all services or portlets have a uniform look and feel – we can ‘skin’ the portlets. The portal also provides seamless transitions from content to content, from services to services, and between content and services.

The following standards are also of interest:

- The Web Services Description Language (WSDL)⁴ standard to provide a model for describing our services;
- The Web Services Business Process Execution Language (WSBPEL)⁵ standard to specify our workflow;
- The Web Services for Remote Portlets (WSRP)⁶ standard to provide an interface for interacting with presentation-oriented services; and
- The Web Services Distributed Management (WSDM)⁷ standard to provide an interface for training providers to manage and monitor the status of the services.
- The Web Services Resource Framework (WSRF)⁸ standard to provide our services with a set of operations so that they become stateful;

Where possible, our services will be described using WSDL, orchestrated using WSBPEL, brokered through WSRP and monitored using WSDM. We can also use WSRF to attach state to our services that can be persisted between user sessions.

4.1 The Models

When authoring an adaptive simulation using the GRAPPLE tool set (see deliverable D3.5a and the definitions in Annex A), there are a variety of models, e.g. the RM, the DM, a model of PRTs, the AM, etc. Annexes B, C, D, and E provide formal XML Schemas for the three main models: the DM, the PRT and the AM. Annexes F, G, and H provide example XML files that are valid with respect to these schemas. They illustrate the structure and typical contents of these models.

These models constitute GRAPPLE’s vision of the pertinent artefacts used to compose an adaptive course or adaptive simulation. However, an adaptive engine can be more versatile than this; it can access any number of models. Furthermore, the purpose and theory behind some newer models might have been unknown at the time the engine was developed. For example, AHA! [17] is capable of delivering adaptive courses composed using a variety of authoring tools.

Our infrastructure is independent of the actual choice of models and tool set used when authoring an adaptive course or adaptive simulation. Of course, it is being developed side-by-side with the GRAPPLE tool set (deliverable D3.5a), but it is not bound to it. To this end, the heterogeneous set of models appear homogeneous to the adaptive engine. They all provide a uniform interface to query and access their content. Some are read-only, for example, the RM and the DM, whereas others are read-write, for example, the UM. To illustrate this more clearly, suppose an author, when creating an adaptive course, develops a new type of PRT. He publishes the course, thereby translating the AM into GAL code that can be executed by an adaptive engine. The translation process also retrieves and embeds this definition of the new PRT. The GAL code has no primitive construct for the PRT itself. It operates at a lower semantic level. It describes the PRT and its adaptive behaviour using more primitive constructs. It accesses models, for example, the UM, in the same way it accesses any other model.

An obvious interface to employ when accessing and modifying these models is a graph-based structure, much like the Document Object Model (DOM) API for valid HTML and well-formed XHTML documents or the Resource Description Framework (RDF) for conceptual modelling of information that is provided through web resources. We can traverse the models, query their ancestors, descendants and siblings, and modify values that we have access to. It also provides an intuitive mental map of a model.

⁴ <http://www.w3.org/TR/wsd/>

⁵ <http://www.oasis-open.org/specs/#wsbpelv2.0>

⁶ <http://www.oasis-open.org/specs/#wsrpv1.0> and <http://www.oasis-open.org/specs/#wsrpv2.0>

⁷ <http://www.oasis-open.org/specs/#wsdmv1.1>

⁸ <http://www.oasis-open.org/specs/#wsrfv1.2>

The models are quite separate from the ‘intelligence’ of an adaptive simulation. The intelligent or dynamic component is specified in GAL whereas the static content is held in the models. This separation enables reuse and interoperability between adaptive simulations.

4.2 The Workflow Engine

The workflow engine orchestrates the delivery of an adaptive simulation. It provides an answer to “what happens next”. In adaptive non-simulated learning environments, there is no need for workflow. If the application is based on adaptive hypermedia, the user clicks on a link and the system simply forwards them to the next page. The actual page that the user is forwarded to may be adapted in some way, but the act of forwarding remains unchanged. In an adaptive simulated learning environment, the user can do much more than simply click on a link. He might close a service, maximize a service, complete some task, allow some time interval to elapse, etc. In some senses, adaptive simulations provide an alternative way to approach adaptive navigation. Instead of content-driven navigation, adaptive simulations allow learners and authors to think in terms of tasks that need to be completed. This may prove to be a more suitable and natural abstraction for authors when designing a course. Also, since the user is learning not just the content and functionality available in each service, but also the way in which the services are combined and sequenced, there is a need for a workflow engine. To make this aspect of the learning process more transparent, a workflow outline is made available through the workflow engine. This is a ‘map’ of the simulation, as being performed by the user and is ultimately derived from the author’s description of the adaptive simulation. It is equivalent to the interpretation of the concepts in an adaptive non-simulated course to provide a structured navigational interface. An aggregated list provides learners with a convenient list of pending tasks across all their adaptive simulations.

Of course, the workflow itself can be adapted. The sequencing of the services may change depending on the user’s role, level of experience, etc. Notable approaches to integrating adaptation with workflow include:

- Event-oriented workflow adaptation [20]: Due to unforeseen circumstances, workflow instances may become inappropriate. For example, a workflow definition may specify the execution of `serviceA`. However, after a workflow instance based on this definition has been started, it may be detected that `serviceA` is no longer available or the user no longer has access to it. This requires that all invocations of `serviceA` be dropped from or replaced in the workflow instance.
- Context-oriented workflow adaptation [6]: Different services may be required in different contexts. However, specifying all possible contexts in a workflow can result in complicated and unwieldy definitions that are difficult to understand and maintain. A simple abstraction based on a hierarchy of different services can address both problems. Basically, the more specific the context, the further down the hierarchy system must traverse in order to find an appropriate service.

We are, in some ways, following the latter approach. The adaptive engine maps portions of the GAL code to an appropriate workflow definition language, e.g. WSBPEL (see Figure 3). However, the adaptive engine can interject and change the workflow as the simulation proceeds. This is markedly different to the approach above, in which the entire workflow definition is expanded at the beginning to cover all possible adaptations and then left to the workflow engine for execution [6]. The problem with this approach is that the workflow outline, as presented to the user, becomes too complicated. They only need to see the workflow that applies to their role and situation. The ‘map’ must reflect the adaptation.

Introducing a workflow engine into the fray brings with it some other important considerations. In adaptive hypermedia, it is assumed that the user is navigating along one path at a time. They can jump to other parts of the hyperspace but they always have a unique single position, i.e. the page they are currently looking at. Although, it is possible that the user could open multiple browsers at the same time and navigate multiple parts of the hyperspace simultaneously, this can be modelled as a user who is jumping back and forth between locations. On the other hand, in adaptive simulations and workflows in general, there may be many parallel executions of workflow. To deliver adaptive simulations we need to handle the concurrent execution of workflow. We need the following components:

- A `split` construct that allows thread-based or process-based execution of workflow;
- A `join` construct that allows synchronization of concurrent executions; and
- Time handling.

Synchronizers allow tasks executing in parallel threads to merge at a synchronization point or to exchange data at a point of time and have critical sections that must be made thread-safe when parallel threads try to execute them simultaneously.

The workflow engine manages the workflow through its lifecycle. At run-time, the workflow engine needs to take care of such things as execution of workflow involving parallel paths, their merging, preservation of state with context, asynchronous and synchronous execution of sub-workflows, and inbound calls to services in the workflow from outside and from other services.

Each workflow instance represents a unique run-time occurrence of the workflow based on a specific workflow invocation (instantiation) request. The state and context of each workflow instance is stored in the workflow database at logical points during the workflow execution. An executable workflow definition is typically in a standard format such as the Business Process Modelling Language (BPML), WSBPEL [31], XPlan [24], or WSPlan [37].

A workflow can contain any mix of services. When the workflow is executed, its services are executed in the order specified in the workflow definition. The execution proceeds until a service that requires interaction with the user is encountered, at which point the workflow engine waits for the user to provide some input. Once the user completes the action, the workflow resumes. Such a workflow would pause at all points where user interaction is involved in the workflow. Also, the BPML constructs such as `All` (`Flow` in WSBPEL) result in a split (fork) and join (merge) having to be supported by the workflow engine.

4.2.1 Starting a Workflow

Each workflow instance is an instance of a thread or process that represents the run-time version of the workflow. The workflow engine creates this object once a request for the execution of the workflow arrives. It contains a list of services in a tree structure that mimics the service order and nesting in the workflow definition and it executes the services by moving through this list.

4.2.2 Parallel Branching

The `All` service in BPML is a complex service; it is composed of one or more other services. An `All` service executes all the services that it is composed of in parallel. The equivalent construct in WSBPEL is `flow`. We can see `All` in terms of a split and join, i.e. the main workflow execution path splits into parallel paths – as many paths as there are services in `All` – executes them in parallel, and once the parallel paths complete execution, they join together with the main workflow path.

The join is a synchronization point that the parallel path threads converge on. The threads reaching this point first wait until others join them and they all synchronize at this point with the main workflow thread. Then their executions end, and the main path will continue thereafter with the workflow execution moving to the next service after the `All`.

4.2.3 Workflow State

The state (including context) of the executing workflow is persisted in the workflow database by the workflow engine at regular logical intervals such as at the start of the workflow, before the execution of services, and after execution of services. The workflow variables and service states are captured along with relevant timestamps. The workflow engine uses the state information to faithfully restore or revive a workflow instance.

Since a workflow can involve parallel paths (split and join) and given that services executing in parallel can update the same variables, whenever the workflow state is stored it has to be done in such a way that the context is logically consistent across services. This functionality will need to be provided by the workflow engine.

4.2.4 Interruptible Services

There may be services in the workflow that are expected to receive invocations into them from outside or from services in other paths of the workflow; these are called *interruptible services*. The incoming invocation contains the input parameters for the service from the caller. Any outputs from the interruptible service are sent back to the caller directly during the time of exchange. Once this is done, the execution of the workflow then proceeds separately in its own thread and the caller's execution workflow separately in the caller thread.

In BPML, a service that is defined with a child element `<input>` followed by an optional `<output>` is considered an interruptible service. Input and output elements define the input and output parameters for the service. This means that this type of service receives invocations, i.e. the service responds to an input message. When workflow execution control reaches this service, the workflow has to wait for an external

party to invoke this action. Once the invocation is received, the service execution proceeds and reaches the next service in the workflow. The equivalent for this in WSBPEL is `receive`. In the case of the other type of service, i.e. where we have `<output>` followed by `<input>`, the workflow engine invokes the action corresponding to the service and so no waiting is involved. Once the workflow instance pauses for the message to arrive, it waits for a configured amount of time, similar to a user service.

An important concept in interruptible services is *correlation*. This refers to the method of mapping the incoming message to the interruptible service of the right workflow instance among a set of running workflow instances. We need a unique identifier in the incoming message.

4.3 The Adaptive Engine

A cornerstone of our infrastructure is the adaptive engine, or GALE (see D1.3a). This is responsible for interpreting and enacting the description of an adaptive simulation, reconciling the various models, and controlling the workflow engine. It can interpret GAL code and execute the appropriate behaviour. It can access, query, and modify the models to retrieve the current resource instances, update the user model, etc. It can produce a workflow definition for the workflow engine and interject when it needs to be adapted. The adaptive engine is surrounded by the other components (see Figure 3) and is accessed directly by neither the author nor the learner.

For the purposes of an adaptive simulation, we use a more general adaptive engine than that for delivering adaptive hypermedia. It accesses new models (e.g. the SDM) and provides for services and adaptive workflow between services. KnowledgeTree [10] is an architecture for developing component-based adaptive e-Learning using distributed reusable intelligent learning activities. However, it focuses on intelligent content, that is, content that can be invoked as a service, but it does not support any form of information flow between those services. On the other hand, workflow engines and workflow authoring tools are entirely service-centric and do not provide for content. We need a combination of the two. The adaptive engine, in combination with the workflow engine and portal, provides this.

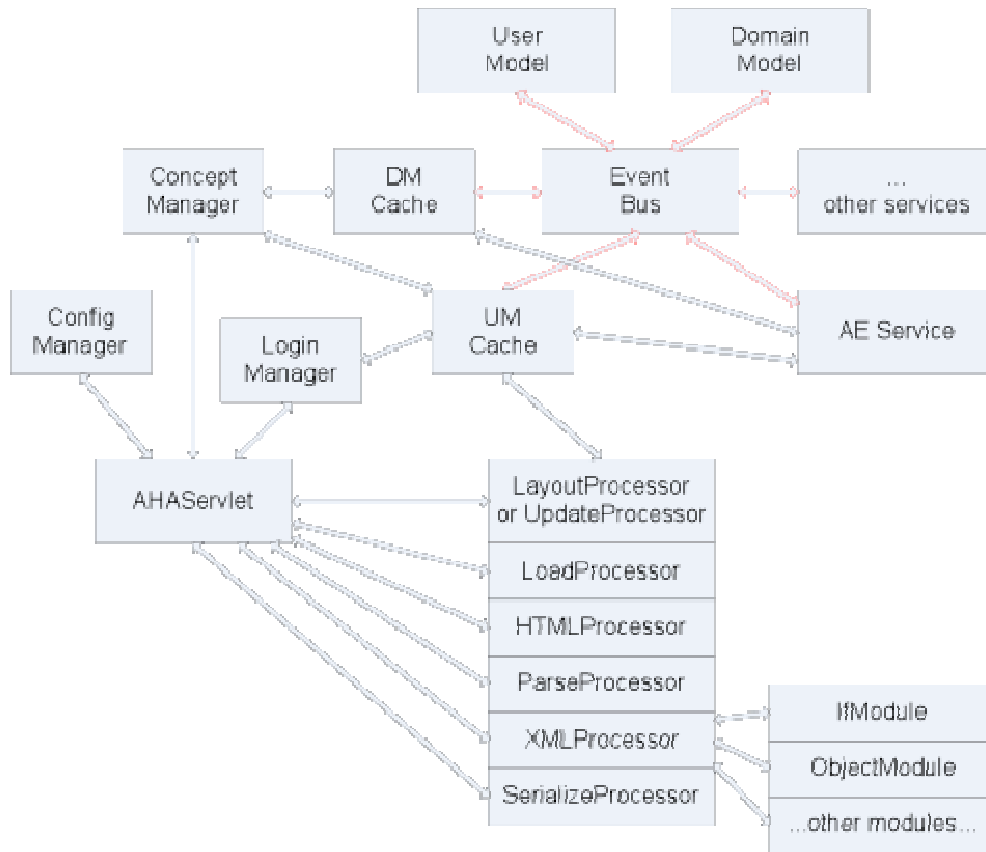


Figure 5:GALE's architecture for the GRAPPLE project. (Reproduced from deliverable D1.3a.)

AHA! [17], as it stands, focuses on content adaptation and does not support services. However, Work Package 1 is extending it within the GRAPPLE project. GALE (the adaptive engine that will supersede AHA!) will be able to support GAL (see Section 4.6). This will include any simulation-specific functionality added to GAL. We can communicate with GALE through the event bus (see Figure 5). Components send requests to the event bus, and other components listen to these events and handle them if they so wish. In particular, the workflow engine and portal will communicate with the adaptive engine through this interface (see deliverable D1.3a for more details).

4.4 The Portal

A portlet is a software component with a user interface that is contained in a portlet container and displayed (and possibly aggregated with other portlets) by a portal. A portlet can provide, for example, blogging, bookmarking, email, mash-up, or search functionality (see Figure 6). These are the elemental services from which an author can compose an adaptive simulation. Specialized portlets, e.g. *scrumMeeting*, provide for the composition of specialized adaptive simulations, e.g. the Scrum software development process (see deliverable D3.5a for a complete scenario involving this process).



Figure 6: An aggregation of portlets for inspecting financial data (©2009 Google: The iGoogle interface).

Portals often provide system integration and consistency, single sign-on and security, personalization, collaboration, task management and workflow, and internationalization. Portal software abounds, including open-source projects, e.g. Apache Jetspeed, Apache Orchestration Director Engine (ODE) [5], the GridSphere Portal Framework [2], and uPortal [23]; commercial products with open-source editions, e.g. eXo Portal [1], and Liferay Portal [25]; and entirely commercial products, e.g. Oracle's Application Server Portal [35], IBM's WebSphere Portal [22], Microsoft SharePoint Portal Server [28], and Sybase's Enterprise Portal [45].

The provision of adaptive simulations is certainly a demanding portal application. Several portlets need to work together to present learning content and services, adapt the presentation to the learner's knowledge, and monitor the learner's progress. We also need to incorporate workflow into portlets so that we can assign tasks to learners and groups of learners, constrain when a service can be performed and by whom, etc. We integrate a third-party workflow engine within the portal to deliver adaptive simulations.

The portal provides the user interface when delivering adaptive simulations. Any portlet can be minimized, maximized, customized, or closed by the user. Each adaptive simulation is realized through a single portlet. The container or *outer portlet* can instantiate *inner portlets*. These instantiations are determined by the workflow of the adaptive simulation. The inner portlets realize the individual services. Of course, it is entirely possible that the inner portlets can be instantiated outside of the outer portlet by some other application, in which case they are not under the control of the workflow of any adaptive simulation. This is analogous to a user opening a piece of content that is displayed through an adaptive hypermedia application by accessing the resource directly. For our purposes, we require the portlets, portlet containers and portals to adhere to the Portlet Specifications [3][4] and the WSRP Specification [32].

4.5 Author-Time, Publish-Time, and Run-Time

There are three distinct stages that an adaptive simulation passes through. Firstly, an author creates an adaptive simulation at *author-time* using the authoring tools (see deliverable D3.5a). The adaptive simulation is then published at *publish-time*. This involves the translation of the adaptive simulation into a form that can be executed by an adaptive engine. The output of this stage is GAL code (see Section 4.6). Finally, the adaptive simulation is executed at *run-time* by an adaptive engine. An author can maintain or modify an adaptive simulation by re-entering the authoring tools; changes are propagated through publish-time to the form that is executed at run-time to update the adaptive simulation on the fly.

4.6 The GRAPPLE Adaptation Language

GAL expresses adaptive navigation that can be performed by an adaptive engine. Work Packages 1 and 3 are defining this language within the GRAPPLE project. GAL, as it currently stands, is based on Semantic Web specifications and languages. It uses a Turtle-like syntax to construct an RDF graph for each model. GAL's primitives include variables, a set data structure, a representation of 'links', conditionals, and events. Brusilovsky's adaptation techniques [9][12] can be expressed using these primitives. A CRT is a relationship between domain model and user model 'classes' (e.g. gal:Concept, gal:Relationship, gal:Resource, gal:User, gal:HasKnowledge, gal:isSuitable, etc.) that concerns adaptive

navigation. There is also an application model defining a global adaptive navigation structure. 'Active' content is provided through forms with buttons, text inputs, and choice inputs. GAL comprises an adaptation language, query language (using a quasi-SPARQL syntax), and update language.

The simulation component of GRAPPLE will require simulation-specific functionality. We will strive for a complete set of functionality. However, it is reasonable to leave some of the more difficult functionality unimplemented at this stage. We would like GAL to have:

- Basic control flow constructs;
- Basic looping constructs; and
- Event-handling (events are fired by the 'system', GAL code is executed by the adaptive engine in response to these events, then control is returned to the system).

Adaptive features (what we can adapt) can be included as follows:

```
gal.features.hypermedia.content.text.fragments // insertion, removal, alteration
gal.features.hypermedia.content.text.fragmentSorting
gal.features.hypermedia.content.text.conditionalText
gal.features.hypermedia.content.text.frames
gal.features.hypermedia.content.text.stretchText
gal.features.hypermedia.content.text.dimText
gal.features.hypermedia.content.multimedia

gal.features.hypermedia.navigation.linkHiding // disabling, hiding, removal
gal.features.hypermedia.navigation.linkSorting
gal.features.hypermedia.navigation.linkGeneration
gal.features.hypermedia.navigation.linkAnnotation
gal.features.hypermedia.navigation.directGuidance
gal.features.hypermedia.navigation.mapping

gal.features.simulation.procedural.content.serviceSelection
gal.features.simulation.procedural.content.fidelity // increase or decrease
gal.features.simulation.procedural.content.augmentation // includes
instructional support, increase or decrease
gal.features.simulation.procedural.content.modality // what mode of interaction?
e.g. command line, point-and-click etc.
gal.features.simulation.procedural.content.feedback // natural vs. artificial
and immediate vs. delayed
gal.features.simulation.procedural.content.motivators
gal.features.simulation.procedural.navigation.sequencing
gal.features.simulation.procedural.navigation.control // between user and system
gal.features.simulation.procedural.navigation.mapping // based on the workflow
gal.features.simulation.procedural.presentation.arrangement
```

`gal.features.simulation.procedural.content.modality` specifies the mode of interaction (e.g. command line, point-and-click, etc.) and is unique to simulations. Providing several modes within a lesson enhances interest and stimulates learning [41]). The same principle applies when providing types of actions, e.g. making a choice, manipulating an object, reacting to an event, and collecting information.

Adaptive dimensions (what we can adapt to) can also be included as follows:

```
gal.dimensions.individual.goals // aims, goals, tasks, learning outcomes
gal.dimensions.individual.competence // prior knowledge with this application
gal.dimensions.individual.usage // experience with this application
gal.dimensions.individual.traits // learning styles and cognitive styles
gal.dimensions.individual.language
gal.dimensions.individual.background // cultural background, what other systems
have they used?
gal.dimensions.individual.interests
gal.dimensions.individual.milestones // qualifications
gal.dimensions.individual.role
gal.dimensions.individual.motivation

gal.dimensions.group.*

gal.dimensions.system.usage // how our own system is being used as a whole

gal.dimensions.environment.latency
gal.dimensions.environment.context
gal.dimensions.environment.location
gal.dimensions.environment.platform
```

We also need support for events. Events are fired by the system that delivers the content and services, e.g. a hypermedia system (web server), simulation or workflow engine, VR engine, game engine, etc. The adaptive engine in response to these events executes event-handlers, specified in GAL. Typical events include `userLogin(user)`, `userLogout(user)`, `followLink(user, link)`, `openService(user, service)`, and `closeService(user, service)`. These events can be included as follows:

```
gal.events.hypermedia.userLogin // an alias for gal.events.hypermedia.http.post
gal.events.hypermedia.userLogout // an alias for gal.events.hypermedia.http.post
gal.events.hypermedia.followLink // an alias for gal.events.hypermedia.http.get

gal.events.hypermedia.http.get
gal.events.hypermedia.http.post
gal.events.hypermedia.http.put
gal.events.hypermedia.http.delete

gal.events.simulation.procedural.openService // inputs
gal.events.simulation.procedural.closeService // outputs
gal.events.simulation.procedural.interruptService // interruptible services
gal.events.simulation.procedural.stateChange
```

We now present a snippet of GAL code, as we envision it, when specifying adaptive behaviour for simulations. Please note that this language is still being defined within the GRAPPLE project and is likely to change in future versions. In particular, we have not adopted the Turtle-like syntax promoted in deliverable D1.1a. In other words, this example is for illustrative purposes only. We would like to use dynamic fidelity in an adaptive simulation – the level of fidelity is changed based on the learner’s current instructional level [8][18][21][39][40][44]. This is also known as model progression [18][46]. For a novice, initial learning is emphasized (with lower fidelity) and for an advanced learner, transfer is emphasized (with higher fidelity). The GAL code is as follows:

```
on gal.events.simulation.procedural.stateChange {
  def c = gal.dimensions.individual.competence;
  def m = gal.dimensions.individual.milestones;
  def u = gal.dimensions.individual.usage;
  def f = gal.features.simulation.procedural.content.fidelity;
  def a = gal.features.simulation.procedural.content.augmentation;

  // aggregate dimension values to get where the user is now
  def max = c.MAX_VALUE + m.MAX_VALUE + u.MAX_VALUE;
  def min = c.MIN_VALUE + m.MIN_VALUE + u.MIN_VALUE;
  def value = (c.value + m.value + u.value - min) / (max - min) * 100;

  // fidelity increases and augmentation decreases proportionality
  f.value = f.MIN_VALUE + ((f.MAX_VALUE - f.MIN_VALUE) / 100 * value);
  a.value = a.MAX_VALUE - ((a.MAX_VALUE - a.MIN_VALUE) / 100 * value);
}
```

5 Conclusions

In this deliverable we have outlined the design of the infrastructure for the delivery of adaptive simulations. Whereas traditional adaptive hypermedia systems are content-centric and business process management or workflow systems are service-centric, our infrastructure is a combination of the two. It can cater for both content and service-based learning applications. An adaptive simulation, or more specifically, an adaptive procedural simulation is considered to be an assembly of concepts and services, with pedagogical links between them and a workflow specification guiding the execution of the services. The content, the services, and the workflow itself can be adapted to the learners.

To accomplish this feat, our infrastructure is centred on three main components: a workflow engine, a portal, and an adaptive engine. The workflow engine enacts the adaptive simulation. It interprets the sequence in which the services should be presented to the learner. It can also handle notions that are very different from the simple sequencing of pages in an adaptive hypermedia application, for example, parallel branches of control where the learner may have to complete two or more services simultaneously in order to progress in the simulation. The portal is the point of interaction with the learner. It provides a uniform interface to the constituent content and services of an adaptive simulation. For example, it allows a learner to open, close, minimize, and maximize a service or to rearrange their appearance on the screen. It also enables easy integration with LMSs: portlets operate under well-defined specifications [3][4] that are widely supported. Indeed, we utilize the WSRP Specification [32] to broker all interaction with the portlets so that we can examine and record the user’s actions. The adaptive engine is the crux of the infrastructure. It can modify, at run-time, the choice of content and services, the sequencing of the services, the presentation of the portlets to the learner, etc. Whereas the combination of a workflow engine and a portal is fairly straightforward [7], the combination of a workflow engine, a portal, and an adaptive engine has only recently been investigated

[34] and appears much more challenging. In Figure 3 we show these three components mutually communicating in order to deliver adaptive simulations.

We also reviewed some related work from the fields of adaptive hypermedia and AWSC. We extended the set of requirements for adaptive non-simulated learning environments to cater for adaptive simulations. In particular, we outlined additional features that GAL will be required to support in order to adequately express adaptive simulations and their behaviour. We presented some snippets of what GAL may look like and how it can be used to specify adaptive behaviour.

This deliverable, in conjunction with D3.5a (Design of an Adaptive Simulation Authoring Tool), provide the direction and design for the first implementation phase of the adaptive simulations component. The future deliverables D4.1b (Implementation of the Infrastructure for the Adaptive Delivery of Simulations) and D3.5b (Initial Implementation of the Adaptive Simulation Authoring Tool) will report on this work. At present we are experimenting and beginning work with the following toolkits and technologies: We are using ActiveBPEL (<http://www.activevos.com>) as our workflow engine, Glassfish (<http://glassfish.dev.java.net>), OpenPortal (<http://portal.dev.java.net>), and Apache Pluto (<http://portals.apache.org/pluto>) to contain and aggregate the portlets, and GALE as our adaptive engine. This combination of technologies will provide a flexible, reusable architecture for delivering the content and services that make up an adaptive simulation.

References

- [1] eXo Portal. <http://www.exoplatform.com>.
- [2] The GridSphere Portal Framework. <http://www.gridsphere.org>.
- [3] JSR 168: Portlet Specification. <http://jcp.org/en/jsr/detail?id=168>.
- [4] JSR 286: Portlet Specification 2.0. <http://jcp.org/en/jsr/detail?id=286>.
- [5] Apache. Orchestration Director Engine (ODE). <http://ode.apache.org>.
- [6] L. Ardissono, R. Furnari, A. Goy, G. Petrone, and M. Segnan. A Framework for the Management of Context-Aware Workflow Systems. In Proceedings of the 3rd International Conference on Web Information Systems and Technologies (WEBIST'07), pages 80-87, 2007.
- [7] M. Askren and M. Arseniev. Combating Stovepipes: Implementing Workflow in uPortal. EDUCAUSE Quarterly, 27(2), 2004.
- [8] J. Bruner. Toward a Theory of Instruction. Belknap Press, 2004.
- [9] P. Brusilovsky. Adaptive Hypermedia. User Modeling and User Adapted Interaction, 11(1-2):87-110, 2001.
- [10] P. Brusilovsky. KnowledgeTree: A Distributed Architecture for Adaptive E-Learning. In S. Feldman, M. Uretsky, M. Najork, and C. Wills, editors, Proceedings of the 13th International Conference on World Wide Web – Alternate Track Papers and Posters (WWW'04), pages 104-113. ACM, 2004.
- [11] P. Brusilovsky. Student Model Centred Architecture for Intelligent Learning Environment. In Proceedings of the 4th International Conference on User Modeling (UM'94), pages 31-36, 1994.
- [12] P. Brusilovsky. Methods and Techniques of Adaptive Hypermedia. User Modeling and User-Adapted Interaction, 6(2-3):87-129, 1996.
- [13] P. Brusilovsky and M. Maybury. From Adaptive Hypermedia to the Adaptive Web. Communications of the ACM, 45(5):30-33, 2002.
- [14] P. Brusilovsky and P. Miller. Course Delivery Systems for the Virtual University. In T. Tschang and T. Della Senta, editors, Access to Knowledge: New Information Technologies and the Emergence of the Virtual University, pages 167-206. Elsevier, 2001.
- [15] P. Brusilovsky, S. Ritter, and E. Schwarz. Distributed Intelligent Tutoring on the Web. In B. du Boulay and R. Mizoguchi, editors, Artificial Intelligence in Education: Knowledge and Media in Learning Systems, pages 482-489. IOS Press, 1997.
- [16] P. Brusilovsky, J. Eklund, and E. Schwarz. Web-Based Education for All: A Tool for Developing Adaptive Courseware. In H. Ashman and P. Thistlewaite, editors, Proceedings of the 7th International World Wide Web Conference, pages 291-300, 1998.
- [17] P. De Bra, N. Stash, C. Romero, and S. Ventura. Authoring and Management Tools for Adaptive Educational Hypermedia Systems: The AHA! Case Study. In A. Vellido, F. Castro, A. Nebot, F.

Mugica, D. Tedman, and R. Tedman, editors, *Evolution of Teaching and Learning Paradigms in Intelligent Environment*, pages 285-308. Springer, 2007.

- [18] T. de Jong, E. Martin, J. Zamarro, F. Esquembre, J. Swaak, and W. van Joolingen. The Integration of Computer Simulation and Learning Support: An Example from the Physics Domain of Collisions. *Journal of Research in Science Teaching*, 36(5):597-615, 1999.
- [19] DFKI and Saarland University. Activemath. <http://www.activemath.org>.
- [20] A. Dieberger, P. Dourish, K. Höök, P. Resnick, and A. Wexelblat. Social Navigation: Techniques for Building More Usable Systems. *Interactions*, 7(6):36-45, 2000.
- [21] P. Goodyear, M. Njoo, H. Hijne, and J. Berkum. Learning Processes, Learner Attributes and Simulations. *Education and Computing*, 6(3-4):263-304, 1991.
- [22] IBM. WebSphere Portal. <http://www-01.ibm.com/software/websphere/portal>.
- [23] JA-SIG. uPortal. <http://www.uportal.org>.
- [24] M. Klusch, A. Gerber, and M. Schmidt. Semantic Web Service Composition Planning with OWLS-XPlan. In *Proceedings of the 1st International AAAI Fall Symposium on Agents and the Semantic Web*. AAAI Press, 2005.
- [25] Liferay. Portal. <http://www.liferay.com>.
- [26] D. McDermott. Estimated-Regression Planning for Interactions with Web Services. In M. Ghallab, J. Hertzberg, and P. Traverso, editors, *Proceedings of the 6th International Conference on Artificial Intelligence Planning Systems*, pages 204-211. AAAI, 2002.
- [27] S. McIlraith and T. Son. Adapting Golog for Composition of Semantic Web Services. In D. Fensel, F. Giunchiglia, D. McGuinness, and M. Williams, editors, *Proceedings of the 8th International Conference on Principles of Knowledge Representation and Reasoning (KR'02)*, pages 482-496. Morgan Kaufmann, 2002.
- [28] Microsoft. Sharepoint Portal Server. <http://www.microsoft.com/sharepoint>.
- [29] R. Müller, U. Greiner, and E. Rahm. AGENTWORK: A Workflow-System Supporting Rule-Based Workflow Adaptation. *Data and Knowledge Engineering*, 51(2):223-256, 2004.
- [30] W. Nejd, B. Wolf, S. Staab, and J. Tane. EDUTELLA: Searching and Annotating Resources within an RDF-Based P2P Network. In M. Frank, N. Noy, and S. Staab, editors, *Proceedings of the International Workshop on the Semantic Web (WWW'02)*, 2002.
- [31] OASIS. Web Services Business Process Execution Language (WSBPEL). Technical report. <http://docs.oasis-open.org/wsbpel>.
- [32] OASIS. Web Services for Remote Portlets (WSRP). Technical report. <http://docs.oasis-open.org/wsrp>.
- [33] T. Oda, H. Satoh, and S. Watanabe. Searching Deadlocked Web Learners by Measuring the Similarity of Learning Activities. In *Proceedings of the WWW-Based Tutoring Workshop at the 4th International Conference on Intelligent Tutoring Systems (ITS'98)*, 1998.
- [34] I. O'Keefe, O. Conlan, and V. Wade. A Unified Approach to Adaptive Hypermedia Personalisation and Adaptive Service Composition. In V. Wade, H. Ashman, and B. Smyth, editors, *Proceedings of the 4th International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH'06)*, pages 303-307. Springer, 2006.
- [35] Oracle. Application Server Portal. <http://www.oracle.com/technology/products/ias/portal/index.html>.
- [36] S. Pathak and P. Brusilovsky. Assessing Student Programming Knowledge with Web-Based Dynamic Parameterized Quizzes. In P. Barker and S. Rebelsky, editors, *Proceedings of the World Conference on Educational Multimedia, Hypermedia and Telecommunications (ED-MEDIA'02)*, pages 1548-1553. AACE, 2002.
- [37] J. Peer. A POP-Based Replanning Agent for Automatic Web Service Composition. In *Proceedings of the 2nd European Semantic Web Conference (ESWC'05)*, pages 47-61. Springer, 2005.
- [38] J. Rao and X. Su. A Survey of Automated Web Service Composition Methods. In J. Cardoso and A. Sheth, editors, *Proceedings of the 1st International Workshop on Semantic Web Services and Web Process Composition (SWSWPC'04)*, pages 43-54. Springer, 2004.

- [39] C. Reigeluth. In Search of a Better Way to Organize Instruction: The Elaboration Theory. *Journal of Instructional Development*, 2(3):8-14, 1979.
- [40] C. Reigeluth and E. Schwartz. An Instructional Theory for the Design of Computer-Based Simulations. *Journal of Computer-Based Instruction*, 16(1):1-10, 1989.
- [41] J. Rigney and K. Lutz. Effect of Graphic Analogies of Concepts in Chemistry on Learning and Attitude. *Journal of Educational Psychology*, 68(3):305-311, 1976.
- [42] A. Rios, E. Millán, M. Trella, J. Pérez, and R. Conejo. Internet Based Evaluation System. In S. Lajoie and M. Vivet, editors, *Artificial Intelligence in Education: Open Learning Environments*, pages 387-394. IOS Press, 1999.
- [43] Soller and A. Lesgold. A Computational Approach to Analysing Online Knowledge Sharing Interaction. In U. Hoppe, F. Verdejo, and J. Kay, editors, *Artificial Intelligence in Education: Shaping the Future of Learning through Intelligent Technologies*, pages 253-260. IOS Press, 2003.
- [44] J. Swaak, W. van Joolingen, and T. de Jong. Supporting Simulation-Based Learning; The Effects of Model Progression and Assignments on Definitional and Intuitive Knowledge. *Learning and Instruction*, 8(3):235-252, 1998.
- [45] Sybase. Enterprise Portal (EP). <http://www.sybase.com>.
- [46] J. van der Meij and T. de Jong. Supporting Students' Learning with Multiple Representations in a Dynamic Simulation-Based Learning Environment. *Learning and Instruction*, 16(3):199-212, 2006.
- [47] G. Weber and P. Brusilovsky. ELM-ART: An Adaptive Versatile System for Web-Based Instruction. *International Journal of Artificial Intelligence in Education*, 12(4):351-384, 2001.
- [48] G. Weber, H. Kuhl, and S. Weibelzahl. Developing Adaptive Internet Based Courses with the Authoring System NetCoach. In P. De Bra, P. Brusilovsky, and A. Kobsa, editors, *Proceedings of the 3rd International Workshop on Adaptive Hypertext and Hypermedia (AH'03)*, pages 222-223. Springer, 2002.
- [49] S. Wilson, K. Blinco, and D. Rehak. An e-Learning Framework: A Summary. In *Proceedings of the Advancing Learning Technology Interoperability Lab Conference (ALT-I-LAB'04)*, 2004.

Annexes

A	DEFINITIONS	ERROR! BOOKMARK NOT DEFINED.
5.1	Resource Model	Error! Bookmark not defined.
5.2	Domain Model	Error! Bookmark not defined.
5.3	Pedagogical Relationship Types	Error! Bookmark not defined.
5.4	Adaptation Model	Error! Bookmark not defined.
5.5	Activities	Error! Bookmark not defined.
B	COMMON XML SCHEMA	ERROR! BOOKMARK NOT DEFINED.
C	XML SCHEMA FOR THE DOMAIN MODEL	ERROR! BOOKMARK NOT DEFINED.
D	XML SCHEMA FOR THE PEDAGOGICAL RELATIONSHIP TYPE	ERROR! BOOKMARK NOT DEFINED.
E	XML SCHEMA FOR THE ADAPTATION MODEL	ERROR! BOOKMARK NOT DEFINED.
F	EXAMPLE XML FILE FOR THE DOMAIN MODEL	ERROR! BOOKMARK NOT DEFINED.
G	EXAMPLE XML FILE FOR THE PEDAGOGICAL RELATIONSHIP TYPE ...	ERROR! BOOKMARK NOT DEFINED.
H	EXAMPLE XML FILE FOR THE ADAPTATION MODEL	ERROR! BOOKMARK NOT DEFINED.

A Definitions

In this annex we present an abbreviated list of definitions that are used in this and related deliverables (D3.5a, D3.5b, D3.5c, and D4.1b) with respect to adaptive simulations. The complete set of definitions can be found in deliverable D3.5a.

5.1 Resource Model

A *Resource Model* (RM) models all assets (concept instances and service instances) that are available for use in an adaptive course or adaptive simulation. These assets comprise text, images, sound, video, chat services, email services, forum services, exercises, references, datasets, etc. The actual assets can be contained in a closed-corpus repository or retrieved from an open-corpus repository, such as the Web. However, every modelled asset must provide meta-data in the form of a required set of pedagogical attributes. These attributes must specify, for instance, the difficulty of the content (e.g. introductory, intermediate, expert), the language, the media format, etc. Service instances also need to expose their inputs and outputs.

One standardized possibility for this task is the IEEE Learning Object Metadata (LOM) specification. This specification defines a number of vocabularies for describing pedagogical attributes of resources. For example, it includes attributes like interactivity type, learning resource type, interactivity level, semantic density, difficulty, and description. IEEE LOM also encapsulates all of the Dublin Core elements. The final choice for the required set of pedagogical attributes has not yet been made.

5.2 Domain Model

A *Domain Model* (DM) comprises a *Concept Domain Model* (CDM) and a *Service Domain Model* (SDM). A SDM can also be considered a *Service Workflow Model* (SWM). The reason for this alternative name is explained below. A CDM models abstract concepts. The actual concept instances are modelled in a RM. D3.1a (Design Specification of a DM Definition Tool) describes this model in more detail. A SDM is an analogous model for services. It models abstract services. The actual service instances are modelled in a RM. Services can be domain specific in their very nature, e.g. `scrumMeeting` is only meaningful when learners are studying the Scrum software methodology, whereas other services can be domain-independent but can be parameterized so that they appear domain-specific, e.g. `classDiscussion` can apply to any domain, but when the discussion is parameterized with a particular topic, it becomes domain-specific.

A CDM will use the IMS Vocabulary Definition Exchange (IMS VDEX) specification (see D3.1a). This specification provides a grammar for the exchange of a common vocabulary. Specifically, VDEX defines a grammar for the exchange of simple machine-readable lists of terms, along some simple semantics to help a human understand the meaning or applicability of the various terms. VDEX may be used to express terms in instances of IEEE LOM, IMS Metadata, IMS LIP, ADL SCORM, etc. VDEX can also express hierarchical and loose networks of relationships if required.

To enable reuse of the DM tool for a SDM, we will also use IMS VDEX to express our 'service space' and some simple descriptive (non-pedagogical) relationships between those services. It will be possible to mix concepts and services together in the one space, blurring the distinction between a CDM and a SDM. The author will simply see a collection of items in a unified DM.

The simple descriptive (non-pedagogical) relationships between services may in some cases be workflow constructs, e.g. XOR-split, XOR-join, etc. These are not describing a workflow or process *per se*. A SDM is a static representation of services. However, if two services are dragged from a SDM into an Adaptation Model (AM) (see below), then the relationship can induce an actual workflow construct if the author so wishes. The relationship in a SDM can be considered a 'default' or 'suggested' workflow construct between two related services. The relationship it induces in an AM is a SRT (again, see below), into which adaptivity can be specified.

Many concept instances in a RM can potentially fulfil the needs of an abstract concept in a CDM. Similarly, many service instances in a RM can potentially fulfil the needs of an abstract service in a SDM. However, for service instances and abstract services, 'fulfilling the needs' is not simply a matter of having appropriate values for certain meta-data attributes. The service instances must also be, to some degree, IOPE-matched, i.e. they must share a similar set of inputs, outputs, preconditions, and effects.

In our authoring tools and delivery infrastructure for adaptive simulations, services are implemented using Portlets and WSRP. This specification allows Portlets to be accessed using Web Services. Portlets have a

well-defined interface and communication protocol so all interaction with them can be brokered. We will require this ability when orchestrating the services through a workflow engine.

Unlike concept instances, service instances can terminate in a variety of ways. These are:

- Through a user action (close service – this is similar to browsing away from a concept instance in a hypertext);
- Through an interrupt from another service instance; or
- Following the expiration of a specified duration (timeout).

5.3 Pedagogical Relationship Types

A *Pedagogical Relationship Type* (PRT) is a *Concept Relationship Type* (CRT), a *Service Relationship Type* (SRT), or a *Concept-Service Relationship Type* (CSRT). CRTs are described in detail in D3.2a (Definition of a Concept Relationship Type Tool). A CRT is a pedagogical relationship between concepts and can be instantiated in an AM (see below), e.g. `prerequisite`, `analogousTo`, etc. It may have associated adaptive behaviour, e.g. only make the target endpoints of a prerequisite CRT visible if its source endpoints have already been visited. Similarly, a SRT is a pedagogical relationship between services and can also be instantiated in an AM. In our implementation, we employ the following seven basic workflow constructs as SRTs: `sequence`, `XOR-split`, `AND-split`, `OR-split`, `XOR-join`, `AND-join`, and `OR-join`. Each of these may also have associated adaptive behaviour. Finally, a CSRT is a pedagogical relationship between concepts and services and can also be instantiated in an AM. Examples include `isExemplifiedBy` (the details of a concept may be explicitly demonstrated by a service), `adheresToThePrinciplesOf` (a service may abide by the rules or laws of a concept), etc.

To the author, all three types of PRT can be mixed together seamlessly in the AM, thereby blurring any distinction between them. However, for us, their functionality will be quite different and so we differentiate between them as such. The authoring tool will provide a repository of commonly used PRTs. D3.2a describes a CRT tool for creating new CRTs from the ground up. This will be extended for creating new SRTs and new CSRTs. A PRT is represented in XML. This XML specifies, among other things, a PRT's name or identifier, description of purpose, arity, parameters, constraints (when and where it can and can't be applied), and its adaptive behaviour (specified in the GAL). This representation will be almost identical to that specified in D3.2a.

5.4 Adaptation Model

An *Adaptation Model* (AM) is a workspace into which everything else is combined to produce an adaptive course or adaptive simulation. Within the GRAPPLE project and D3.3a (Design of a CAM Definition Tool), it is referred to as the *Conceptual Adaptation Model* (CAM). However, this overly emphasizes the role of concepts, especially when the aim is to author adaptive simulations. For this reason we drop the 'conceptual' qualifier in this and related deliverables (D3.5b, D3.5c, D4.1a, and D4.1b) to show that it can also be a service-based AM. The CAM Tool (as described in D3.3a) will be extended to handle services, SRTs, and CSRTs.

PRTs (CRTs, SRTs, and CSRTs) are instantiated in an AM. They are 'dragged' into the workspace by the author and bound to specific concepts and services. A PRT can only be instantiated if the concepts and services that the author is trying to bind it to match its parameters and the PRT's constraints are satisfied. An AM is the author's complete description of an adaptive course or adaptive simulation before it is translated into GAL and enacted by an adaptive engine. An AM is represented in XML. This XML will specify, among other things, the AM's name or identifier, author, and a graph-based model of its contents (concepts, services, and PRTs). This representation will be almost identical to that specified in D3.3a.

5.5 Activities

Activities or *Learning Activities* are simply reusable groupings of concepts, services and PRTs. Examples include `scrumMeetingLearningActivity`, `sprintPlanningMeetingLearningActivity`, `sprintLearningActivity`, and `sprintRetrospectiveMeetingLearningActivity`. Within the AM tool, they can be contracted or expanded to make the visual representation of an adaptive course or adaptive simulation easier to navigate and understand. It is also possible to nest and reuse activities.

B Common XML Schema

```

<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://www.grapple-project.org" xmlns="http://www.grapple-
project.org"
  elementFormDefault="qualified" attributeFormDefault="unqualified">

  <!-- GUIDType -->
  <xs:simpleType name="GUIDType">
    <xs:restriction base="xs:string">
      <xs:pattern
        value="[a-z\d]{8}\-[a-z\d]{4}\-[a-z\d]{4}\-[a-z\d]{4}\-[a-z\d]{12}" />
    </xs:restriction>
  </xs:simpleType>

  <!-- nameType -->
  <xs:simpleType name="nameType">
    <xs:restriction base="xs:string">
      <xs:maxLength value="32" />
    </xs:restriction>
  </xs:simpleType>

  <!-- descriptionType -->
  <xs:simpleType name="descriptionType">
    <xs:restriction base="xs:string">
      <xs:maxLength value="256" />
    </xs:restriction>
  </xs:simpleType>

  <!-- keywordsType -->
  <xs:complexType name="keywordsType">
    <xs:sequence>
      <xs:element name="keyword" minOccurs="0" maxOccurs="unbounded">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:maxLength value="32" />
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>

  <!-- versionType -->
  <xs:complexType name="versionType">
    <xs:sequence>
      <xs:element name="major" type="xs:positiveInteger" />
      <xs:element name="minor" type="xs:nonNegativeInteger" />
    </xs:sequence>
  </xs:complexType>

  <!-- permissionsType -->
  <xs:complexType name="permissionsType">
    <xs:sequence>
      <xs:element name="user">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="read" type="xs:boolean" />
            <xs:element name="write" type="xs:boolean" />
            <xs:element name="publish" type="xs:boolean" />
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>

```

```

<xs:element name="group">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="read" type="xs:boolean" />
      <xs:element name="write" type="xs:boolean" />
      <xs:element name="publish" type="xs:boolean" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="world">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="read" type="xs:boolean" />
      <xs:element name="write" type="xs:boolean" />
      <xs:element name="publish" type="xs:boolean" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>

<!-- pedagogicalAttributesType -->
<xs:complexType name="pedagogicalAttributesType">
  <xs:sequence>
    <xs:element name="difficulty">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:pattern value="beginner/intermediate/advanced" />
        </xs:restriction>
      </xs:simpleType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
</xs:schema>

```

C XML Schema for the Domain Model

```

<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://www.grapple-project.org" xmlns="http://www.grapple-
project.org"
  elementFormDefault="qualified" attributeFormDefault="unqualified">

  <xs:include schemaLocation="grapple_common.xsd"/>

  <!-- entitiesType -->
  <xs:complexType name="entitiesType">
    <xs:sequence>
      <xs:element name="entity" minOccurs="0" maxOccurs="unbounded">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="identifier" type="GUIDType" />
            <xs:element name="name" type="nameType" />
            <xs:element name="description" type="descriptionType" />
            <xs:element name="keywords" type="keywordsType" />
            <xs:element name="resources">
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="resource" minOccurs="1"
maxOccurs="unbounded">
                    <xs:complexType>
                      <xs:sequence>

```

```

        <xs:element name="identifier" type="GUIDType" />
        <xs:element name="pedagogicalAttributes"
type="pedagogicalAttributesType" />
    </xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
<xs:attribute name="type" use="required">
    <xs:simpleType>
        <xs:restriction base="xs:string">
            <xs:pattern value="concept|service" />
        </xs:restriction>
    </xs:simpleType>
</xs:attribute>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>

<!-- relationshipsTypesType -->
<xs:complexType name="relationshipTypesType">
    <xs:sequence>
        <xs:element name="relationshipType" minOccurs="0"
maxOccurs="unbounded">
            <xs:complexType>
                <xs:sequence>
                    <xs:element name="identifier" type="GUIDType" />
                    <xs:element name="label" type="nameType" />
                    <xs:element name="endpoints">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element name="tails" type="xs:nonNegativeInteger" />
                                <xs:element name="heads" type="xs:nonNegativeInteger" />
                                <xs:element name="both" type="xs:nonNegativeInteger" />
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                </xs:sequence>
            </xs:complexType>
        </xs:element>
    </xs:sequence>
</xs:complexType>

<!-- relationshipsType -->
<xs:complexType name="relationshipsType">
    <xs:sequence>
        <xs:element name="relationship" minOccurs="0" maxOccurs="unbounded">
            <xs:complexType>
                <xs:sequence>
                    <xs:element name="identifier" type="GUIDType" />
                    <xs:element name="type" type="GUIDType" />
                    <xs:element name="endpoints">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element name="tails">
                                    <xs:complexType>
                                        <xs:sequence>
                                            <xs:element name="endpoint" type="GUIDType"
minOccurs="0" maxOccurs="unbounded" />
                                        </xs:sequence>
                                    </xs:complexType>
                                </xs:element>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                </xs:sequence>
            </xs:complexType>
        </xs:element>
    </xs:sequence>
</xs:complexType>

```



```

<xs:complexType>
  <xs:sequence>
    <xs:element name="parameter" minOccurs="0" maxOccurs="unbounded">
      <xs:complexType>
        <xs:simpleContent>
          <xs:extension base="xs:string">
            <xs:attribute name="identifier" use="required"
type="GUIDType" />
            <xs:attribute name="mandatory" use="required"
type="xs:boolean" />
          </xs:extension>
        </xs:simpleContent>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
<xs:element name="heads">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="parameter" minOccurs="0" maxOccurs="unbounded">
        <xs:complexType>
          <xs:simpleContent>
            <xs:extension base="xs:string">
              <xs:attribute name="identifier" use="required"
type="GUIDType" />
              <xs:attribute name="mandatory" use="required"
type="xs:boolean" />
            </xs:extension>
          </xs:simpleContent>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="both">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="parameter" minOccurs="0" maxOccurs="unbounded">
        <xs:complexType>
          <xs:simpleContent>
            <xs:extension base="xs:string">
              <xs:attribute name="identifier" use="required"
type="GUIDType" />
              <xs:attribute name="mandatory" use="required"
type="xs:boolean" />
            </xs:extension>
          </xs:simpleContent>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>

<!-- preConstraintsType -->
<xs:complexType name="preConstraintsType">
  <xs:sequence>
    <xs:element name="preConstraint">
      <xs:complexType>
        <xs:choice>
          <xs:element name="minDegree">

```

```

        <xs:complexType>
          <xs:simpleContent>
            <xs:extension base="xs:nonNegativeInteger">
              <xs:attribute name="parameter" use="required"
type="GUIDType" />
            </xs:extension>
          </xs:simpleContent>
        </xs:complexType>
      </xs:element>
      <xs:element name="maxDegree">
        <xs:complexType>
          <xs:simpleContent>
            <xs:extension base="xs:nonNegativeInteger">
              <xs:attribute name="parameter" use="required"
type="GUIDType" />
            </xs:extension>
          </xs:simpleContent>
        </xs:complexType>
      </xs:element>
      <xs:attribute name="identifier" use="required" type="GUIDType" />
    </xs:complexType>
  </xs:element>
</xs:sequence>
</xs:complexType>

<!-- postConstraintsType -->
<xs:complexType name="postConstraintsType">
  <xs:sequence>
    <xs:element name="postConstraint">
      <xs:complexType>
        <xs:choice>
          <xs:element name="minDegree">
            <xs:complexType>
              <xs:simpleContent>
                <xs:extension base="xs:nonNegativeInteger">
                  <xs:attribute name="parameter" use="required"
type="GUIDType" />
                </xs:extension>
              </xs:simpleContent>
            </xs:complexType>
          </xs:element>
          <xs:element name="maxDegree">
            <xs:complexType>
              <xs:simpleContent>
                <xs:extension base="xs:nonNegativeInteger">
                  <xs:attribute name="parameter" use="required"
type="GUIDType" />
                </xs:extension>
              </xs:simpleContent>
            </xs:complexType>
          </xs:element>
        </xs:choice>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>

<!-- inferencesType -->
<xs:complexType name="inferencesType">
  <xs:sequence>
    <xs:element name="inference">

```

```

<xs:complexType>
  <xs:choice>
    <xs:element name="relationshipType" type="xs:string" />
    <xs:element name="relationshipTypeIdentifier" type="GUIDType" />
  </xs:choice>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>

<!-- galType -->
<xs:simpleType name="galType">
  <xs:restriction base="xs:string">
    <xs:maxLength value="1024" />
  </xs:restriction>
</xs:simpleType>

<!-- grapplePRT -->
<xs:element name="grapplePRT">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="identifier" type="GUIDType" />
      <xs:element name="author" type="GUIDType" />
      <xs:element name="name" type="nameType" />
      <xs:element name="description" type="descriptionType" />
      <xs:element name="keywords" type="keywordsType" />
      <xs:element name="version" type="versionType" />
      <xs:element name="created" type="xs:dateTime" />
      <xs:element name="lastUpdated" type="xs:dateTime" />
      <xs:element name="permissions" type="permissionsType" />
      <xs:element name="parameters" type="parametersType" />
      <xs:element name="preConstraints" type="preConstraintsType" />
      <xs:element name="postConstraints" type="postConstraintsType" />
      <xs:element name="inferences" type="inferencesType" />
      <xs:element name="GAL" type="galType" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>

```

E XML Schema for the Adaptation Model

```

<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://www.grapple-project.org" xmlns="http://www.grapple-
project.org"
  elementFormDefault="qualified" attributeFormDefault="unqualified">

  <xs:include schemaLocation="grapple_common.xsd"/>

  <!-- entitiesType -->
  <xs:complexType name="entitiesType">
    <xs:sequence>
      <xs:element name="entity" minOccurs="0" maxOccurs="unbounded">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="pedagogicalAttributes"
type="pedagogicalAttributesType" />
          </xs:sequence>
          <xs:attribute name="identifier" use="required" type="GUIDType" />
          <xs:attribute name="majorVersion" use="required"
type="xs:positiveInteger" />

```

```

        <xs:attribute name="minorVersion" use="required"
type="xs:nonNegativeInteger" />
    </xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>

<xs:complexType name="relationshipsType">
    <xs:sequence>
        <xs:element name="relationship">
            <xs:complexType>
                <xs:sequence>
                    <xs:element name="entitiesToParameters">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element name="entityToParameter" minOccurs="0"
maxOccurs="unbounded">
                                    <xs:complexType>
                                        <xs:sequence>
                                            <xs:element name="entity" type="GUIDType" />
                                            <xs:element name="parameter" type="GUIDType" />
                                        </xs:sequence>
                                    </xs:complexType>
                                </xs:element>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                </xs:sequence>
            </xs:complexType>
            <xs:attribute name="identifier" use="required" type="GUIDType" />
            <xs:attribute name="majorVersion" use="required"
type="xs:positiveInteger" />
            <xs:attribute name="minorVersion" use="required"
type="xs:nonNegativeInteger" />
        </xs:complexType>
    </xs:element>
</xs:sequence>
</xs:complexType>

<!-- grappleAM -->
<xs:element name="grappleAM">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="identifier" type="GUIDType" />
            <xs:element name="author" type="GUIDType" />
            <xs:element name="name" type="nameType" />
            <xs:element name="description" type="descriptionType" />
            <xs:element name="keywords" type="keywordsType" />
            <xs:element name="version" type="versionType" />
            <xs:element name="created" type="xs:dateTime" />
            <xs:element name="lastUpdated" type="xs:dateTime" />
            <xs:element name="permissions" type="permissionsType" />
            <xs:element name="entities" type="entitiesType" />
            <xs:element name="relationships" type="relationshipsType" />
        </xs:sequence>
    </xs:complexType>
</xs:element>
</xs:schema>

```

F Example XML file for the Domain Model

```

<?xml version="1.0" encoding="UTF-8"?>
<grappleDM xmlns="http://www.grapple-project.org"

```

```

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.grapple-project.org
  ../../schemas/grapple_dm.xsd">

<!-- general information -->
<identifier>515f80b9-3ef0-4811-b531-294924a642a2</identifier>
<author>27214759-3a23-41d5-84f8-bddd419483de</author>
<name>Scrum DM</name>
<description>This is a Scrum DM.</description>
<keywords>
  <keyword>scrum</keyword>
</keywords>
<version>
  <major>1</major>
  <minor>5</minor>
</version>
<created>2009-01-01T12:00:00Z</created>
<lastUpdated>2009-01-15T15:30:00Z</lastUpdated>

<!-- permissions -->
<permissions>
  <user>
    <read>true</read>
    <write>true</write>
    <publish>true</publish>
  </user>
  <group>
    <read>true</read>
    <write>true</write>
    <publish>true</publish>
  </group>
  <world>
    <read>true</read>
    <write>true</write>
    <publish>true</publish>
  </world>
</permissions>

<!-- entities -->
<entities>
  <entity type="concept">
    <identifier>cf5de7f5-12b5-4720-92e5-736cac59985b</identifier>
    <name>Scrum Meeting</name>
    <description>This is the concept of a Scrum Meeting.</description>
    <keywords>
      <keyword>scrum</keyword>
      <keyword>meeting</keyword>
    </keywords>
    <resources>
      <resource>
        <identifier>3196d7ad-21f9-4af9-866f-d2c7269a8857</identifier>
        <pedagogicalAttributes>
          <difficulty>beginner</difficulty>
        </pedagogicalAttributes>
      </resource>
      <resource>
        <identifier>266c9ece-e1d0-443b-8457-454587acff1e</identifier>
        <pedagogicalAttributes>
          <difficulty>intermediate</difficulty>
        </pedagogicalAttributes>
      </resource>
      <resource>
        <identifier>843919e9-5b31-4392-841d-274df1b2e980</identifier>

```

```

    <pedagogicalAttributes>
      <difficulty>advanced</difficulty>
    </pedagogicalAttributes>
  </resource>
</resources>
</entity>
<entity type="service">
  <identifier>dae895ee-493e-4bfa-85e7-7e19957d2d7c</identifier>
  <name>Scrum Meeting</name>
  <description>This is the service of a Scrum Meeting.</description>
  <keywords>
    <keyword>scrum</keyword>
    <keyword>meeting</keyword>
  </keywords>
  <resources>
    <resource>
      <identifier>332da5fd-40bb-47ec-9cd2-502f09601d29</identifier>
      <pedagogicalAttributes>
        <difficulty>beginner</difficulty>
      </pedagogicalAttributes>
    </resource>
    <resource>
      <identifier>ea6de0a7-43c2-4936-b9f4-3e928710485d</identifier>
      <pedagogicalAttributes>
        <difficulty>intermediate</difficulty>
      </pedagogicalAttributes>
    </resource>
    <resource>
      <identifier>65126d0e-a547-453c-8422-6bb5f1035e64</identifier>
      <pedagogicalAttributes>
        <difficulty>advanced</difficulty>
      </pedagogicalAttributes>
    </resource>
  </resources>
</entity>
</entities>

<!-- relationship types -->
<relationshipTypes>
  <relationshipType>
    <identifier>13d83413-b6c5-4b4a-a1c8-f5c65c617824</identifier>
    <label>is-a</label>
    <endpoints>
      <tails>1</tails>
      <heads>1</heads>
      <both>0</both>
    </endpoints>
  </relationshipType>
  <relationshipType>
    <identifier>a6781714-fce8-4af6-9d45-4b97a7356a6b</identifier>
    <label>has-a</label>
    <endpoints>
      <tails>1</tails>
      <heads>1</heads>
      <both>0</both>
    </endpoints>
  </relationshipType>
  <relationshipType>
    <identifier>bb69652e-efdc-406e-8021-bd4c50369ecc</identifier>
    <label>implements-a</label>
    <endpoints>
      <tails>1</tails>
      <heads>1</heads>
    </endpoints>
  </relationshipType>

```

```

    <both>0</both>
  </endpoints>
</relationshipType>
</relationshipTypes>

<!-- relationships -->
<relationships>
  <relationship>
    <identifier>7857d980-adf1-4d88-a136-df6fd513a48b</identifier>
    <type>bb69652e-efdc-406e-8021-bd4c50369ecc</type>
    <endpoints>
      <tails>
        <endpoint>dae895ee-493e-4bfa-85e7-7e19957d2d7c</endpoint>
      </tails>
      <heads>
        <endpoint>cf5de7f5-12b5-4720-92e5-736cac59985b</endpoint>
      </heads>
      <both />
    </endpoints>
  </relationship>
</relationships>
</grappleDM>

```

G Example XML file for the Pedagogical Relationship Type

```

<?xml version="1.0" encoding="UTF-8"?>
<grapplePRT xmlns="http://www.grapple-project.org"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.grapple-project.org
  ../../schemas/grapple_prt.xsd">

  <!-- general information -->
  <identifier>e4f90979-ae32-4c9c-b1f2-4f316228f05f</identifier>
  <author>27214759-3a23-41d5-84f8-bddd419483de</author>
  <name>Scrum PRT</name>
  <description>This is a Scrum PRT.</description>
  <keywords>
    <keyword>scrum</keyword>
  </keywords>
  <version>
    <major>2</major>
    <minor>0</minor>
  </version>
  <created>2009-01-01T12:00:00Z</created>
  <lastUpdated>2009-01-15T15:30:00Z</lastUpdated>

  <!-- permissions -->
  <permissions>
    <user>
      <read>true</read>
      <write>true</write>
      <publish>true</publish>
    </user>
    <group>
      <read>true</read>
      <write>true</write>
      <publish>true</publish>
    </group>
    <world>
      <read>true</read>
      <write>true</write>
      <publish>true</publish>

```

```

    </world>
  </permissions>

  <!-- parameters -->
  <parameters>
    <tails>
      <parameter identifier="4b5ce398-fd05-4900-88e8-7f3172ec7fa3"
mandatory="true">parameter_1</parameter>
    </tails>
    <heads>
      <parameter identifier="58aae491-344c-4c17-9400-dfdd063aa0b4"
mandatory="true">parameter_2</parameter>
    </heads>
    <both>
    </both>
  </parameters>

  <!-- pre-constraints -->
  <preConstraints>
    <preConstraint identifier="e4de9a8f-49e2-4852-9ced-6b6e4eca0d64">
      <minDegree parameter="58aae491-344c-4c17-9400-dfdd063aa0b4">0</minDegree>
    </preConstraint>
  </preConstraints>

  <!-- post-constraints -->
  <postConstraints>
    <postConstraint identifier="69b41e75-bda4-4063-8884-fc256fbdcc13">
      <maxDegree parameter="58aae491-344c-4c17-9400-dfdd063aa0b4">1</maxDegree>
    </postConstraint>
  </postConstraints>

  <!-- inferences -->
  <inferences>
    <inference>
      <relationshipType>implements-a</relationshipType>
    </inference>
  </inferences>

  <!-- GAL -->
  <GAL>
    <![CDATA[
      GAL code goes here!
    ]]>
  </GAL>
</grapplePRT>

```

H Example XML file for the Adaptation Model

```

<?xml version="1.0" encoding="UTF-8"?>
<grappleAM xmlns="http://www.grapple-project.org"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.grapple-project.org
../../schemas/grapple_am.xsd">

  <!-- general information -->
  <identifier>24990964-1bb8-48f0-b9fc-5ede90193c9b</identifier>
  <author>27214759-3a23-41d5-84f8-bddd419483de</author>
  <name>Scrum AM</name>
  <description>This is a Scrum AM.</description>
  <keywords>
    <keyword>scrum</keyword>
  </keywords>

```

```

<version>
  <major>1</major>
  <minor>5</minor>
</version>
<created>2009-01-01T12:00:00Z</created>
<lastUpdated>2009-01-15T15:30:00Z</lastUpdated>

<!-- permissions -->
<permissions>
  <user>
    <read>true</read>
    <write>true</write>
    <publish>true</publish>
  </user>
  <group>
    <read>true</read>
    <write>true</write>
    <publish>true</publish>
  </group>
  <world>
    <read>true</read>
    <write>true</write>
    <publish>true</publish>
  </world>
</permissions>

<!-- entities -->
<entities>
  <entity identifier="cf5de7f5-12b5-4720-92e5-736cac59985b" majorVersion="1"
minorVersion="5">
    <pedagogicalAttributes>
      <difficulty>intermediate</difficulty>
    </pedagogicalAttributes>
  </entity>
  <entity identifier="dae895ee-493e-4bfa-85e7-7e19957d2d7c" majorVersion="1"
minorVersion="5">
    <pedagogicalAttributes>
      <difficulty>intermediate</difficulty>
    </pedagogicalAttributes>
  </entity>
</entities>

<!-- relationships -->
<relationships>
  <relationship identifier="e4f90979-ae32-4c9c-b1f2-4f316228f05f"
majorVersion="2" minorVersion="0">
    <entitiesToParameters>
      <entityToParameter>
        <entity>cf5de7f5-12b5-4720-92e5-736cac59985b</entity>
        <parameter>58aae491-344c-4c17-9400-dfdd063aa0b4</parameter>
      </entityToParameter>
      <entityToParameter>
        <entity>dae895ee-493e-4bfa-85e7-7e19957d2d7c</entity>
        <parameter>cf5de7f5-12b5-4720-92e5-736cac59985b</parameter>
      </entityToParameter>
    </entitiesToParameters>
  </relationship>
</relationships>
</grappleAM>

```