

GRAPPLE

3.3a Version: 1.0

Design of CAM

Document Type	Deliverable
Editor(s):	Maurice Hendrix (Warwick), Alexandra Cristea (Warwick)
Author(s):	Maurice Hendrix (Warwick), Alexandra Cristea (Warwick), Martin Harrigan (TCD), Vincent Wade (TCD), Frederic Kleinermann (VUB), Olga De Troyer (VUB)
Reviewer(s):	Eelco Herder (LUH), Ricardo Mazza (USI)
Work Package:	WP3
Due Date:	31-01-2009
Version:	1.0
Version Date:	03-03-2009
Total number of pages:	52

Abstract: This deliverable outlines the definition and design of the Conceptual Adaptation Model (CAM) and CAM tool. The CAM tool is a tool for creating CAM instances with the help of the DM tool and CRT tool (see deliverables 3.1a and 3.2a). The objective of the CAM is to be a system-independent high-level model and tool for straightforward, intuitive graphical authoring of Adaptive Hypermedia. In this document, relations with the DM (Domain Model) and CRT (Concept Relationship Type) models and tools are investigated and a *design of the CAM model, CAM tool and the shell* that links the three tools together is given.

Keyword list: Authoring, Conceptual Adaptation Model, CAM, CAM Tool, CAM Languages, Concept Relationship Types, CRT Tool, Domain Model

Summary

This deliverable outlines the definition and design of the Conceptual Adaptation Model (CAM) and CAM tool.

The objective of the *CAM model* is to be a *system-independent high-level model for straightforward, intuitive graphical authoring* of Adaptive Hypermedia. The objective of the *CAM tool* is to be a tool that illustrates the CAM model, thus allowing for system-independent, high-level modelling for straightforward, intuitive graphical authoring of Adaptive Hypermedia.

The CAM tool is thus a tool for creating CAM instances with the help of the DM tool (as described in deliverable 3.1a) and the CRT tool (described in deliverable 3.2a). In this document relations with the DM and CRT models and tools are investigated and a design of the CAM model, CAM tool and the shell that links the three tools together is given.

The nature of the CAM model and its relation to the DM and the CRT models mean that the Conceptual Adaptation Model is formed of three sub-models, as shall be further detailed in this deliverable:

1. the graphical representation (thus graphical language) of the Conceptual Adaptation Model (the *CAM graphical language*). This language expresses how the domain concepts and CRT instances are appearing on the author's screen.
2. the generic, XML-based representation of the Conceptual Adaptation Model (the *CAM internal language*). This language is the XML-equivalent of the CAM graphical language, and describes how domain concepts and CRT instances are linked together, as well as how they are graphically represented in the author's interface.
3. the generic, XML-based export language of the Conceptual Adaptation Model (the *CAM external language*). This language describes only how domain concepts and CRT instances are linked together, without any graphical information. The CAM external language cannot (normally) be interpreted by an engine by itself, but needs to be paired with the CRT language.

Please note that all these languages are to be generic in the sense of being interpretable by any adaptation-engine, thus they should not be engine-specific. Depending on the semantics of the languages, they can range between *upper-middle* and *lower-middle*, as follows. Languages that express why certain actions should be performed are of upper-middle level. Languages which express what actions should be performed are middle-middle. Languages that express how such actions should be performed belong to the lower middle level.

The remainder of this deliverable is organized as follows: Section 1 shortly introduces the need of the CAM model, based on previous adaptive hypermedia models, and discusses issues which are not fulfilled by these models, leading to the CAM requirements. This section also discusses relations to other parts of the GRAPPLE tool set, and related work. Section 2 extracts more CAM requirements, based on multiple scenarios of desired CAM tool use. Section 3 extracts other requirements, based on model verification considerations, and integrated authoring experience, as well as a few supplementary requirements. Section 4 proposes a design of the CAM model and tool based on the previous requirements: the CAM languages (with more details in Appendix 3). The section also proposes the GRAPPLE shell. More details on the shell are given as Appendix 1, for ease of reading of the main text. Section 5 discusses extensions to the CAM tool, for adaptive simulation and virtual reality. Section 6 concludes. A comprehensive list for the overview of all requirements is given as Appendix 2. A list of all main acronyms and abbreviations is provided at the front of the document.

Authors

Person	Email	Partner code
Maurice Hendrix	maurice@dcs.warwick.ac.uk	WARWICK
Alexandra Cristea	acristea@dcs.warwick.ac.uk	WARWICK
Martin Harrigan	martin.harrigan@cs.tcd.ie	TCD
Vincent Wade	vincent.wade@cs.tcd.ie	TCD
Frederic Kleinermann	frederic.kleinermann@vub.ac.be	VUB
Olga De Troyer	Olga.DeTroyer@vub.ac.be	VUB

Table of Contents

1	INTRODUCTION	7
1.1	The CAM model	8
1.1.1	AHAM	8
1.1.2	LAOS	8
1.1.3	CAM	9
1.2	Relations between CRT tool and other relevant tools of the GRAPPLE project	10
1.2.1	Relation to the DM tool	10
1.2.2	Relation to the CRT tool	10
1.2.3	Relation to GRAPPLE Authoring Shell	10
1.2.4	Relation to Adaptation Engine	11
1.3	Related work	12
2	SCENARIOS OF CAM (TOOL) USE	12
2.1	A generic illustrative scenario	12
2.2	Pedagogical strategies in the CAM tool	15
3	OTHER REQUIREMENTS	20
3.1	Model verification	20
3.1.1	Termination Problems	20
3.1.2	Confluence Problems	21
3.1.3	CRT and DM combination Problems	21
3.1.4	Conclusions on Model verification	21
3.2	Other requirements	22
3.3	Integrated authoring experience: the Shell	22
4	DEFINITION AND DESIGN OF THE CAM & CAM TOOL	23
4.1	CAM Tool architecture	23
4.2	The CAM languages	23
4.2.1	CAM visual language	24
4.2.2	CAM internal language	25
4.2.3	CAM external language	25
4.3	Definition and design of the GRAPPLE Authoring Shell	26
5	EXTENSIONS TO THE CAM TOOL	27
5.1	Adaptive Simulations	27
5.1.1	Simulation-based extensions to the Adaptation Model	27
5.1.2	Simulation-based extensions to the Adaptation Model Tool	29
5.1.3	A simulation-oriented illustrative scenario	29
5.2	Virtual Reality	30
5.2.1	A VR-based illustrative scenario	30
6	CONCLUSIONS	32

REFERENCES 32

THE SHELL..... 35

Shell class 35

Config file 35

AbstractTool..... 35

COMPREHENSIVE LIST OF ALL REQUIREMENTS 37

CAM LANGUAGES SCHEMAS AND EXAMPLES..... 39

1.1 The CAM internal language 39

 1.1.1. Schema..... 39

 3.1.2 Example of Using the Internal CAM XML Schema..... 43

1.2 The CAM external language..... 47

 3.2.1 Schema..... 47

 3.2.2 Example of the Application of the External CAM XML Schema..... 50

Tables and Figures

List of Figures

Figure 1: Information flow of GALE (Grapple Adaptation Engine) components and its input from the CAM (and related models)..... 11

Figure 2. GRAPPLE Software Architecture (taken from deliverable D3.1a)..... 12

Figure 3 A CRT in the CAM tool: here, a prerequisite relation, with empty sockets (placeholders) for concepts..... 13

Figure 4. An instantiated CRT in the CAM tool: here, the Relation between Michelangelo and The Last Judgment..... 13

Figure 5. Relation between Michelangelo and Placeholder Concept _X..... 14

Figure 6. The placeholder represents all concepts for which the creator is Michelangelo..... 14

Figure 7. Another non-instantiated CRT in the CAM tool: Relationship (CRT) for generalization of the Michelangelo example..... 14

Figure 8. Constraints for generalization of the Michelangelo example..... 15

Figure 9. ‘Showafter’ relationship; as visualized in the CAM tool after dragging/selecting it from the CRT instance list; the CRT is not yet instantiated (no DM concepts appear in the Source and Target sockets)..... 16

Figure 10. ‘Showatmost’ via hide relation (only needed if prerequisite does not hide concepts); as visualized in the CAM tool after dragging/selecting it from the CRT instance list; the CRT is not yet instantiated (no DM concepts appear in the Source and Target sockets)..... 16

Figure 11. ‘Countaccess’ relationship; as visualized in the CAM tool after dragging/selecting it from the CRT instance list; the CRT is not yet instantiated (no DM concepts appear in the Source and Target sockets)..... 16

Figure 12. The main relation implementing the ‘Depth First’, the logic in the constraint takes care of showing the appropriate next concept, either the next child or the next sibling; , as visualized in the CAM tool after dragging/selecting it from the CRT instance list; the CRT is instantiated (the socket contains concepts c1, c2, c3, c4 and c5)..... 17

Figure 13. The relation shows $_Y$ if $_X$ has been shown the condition is: $_Y.parent==_X$, as visualized in the CAM tool after dragging/selecting it from the CRT instance list; the CRT is not yet instantiated (no DM concepts appear in the Source and Target sockets)..... 18

Figure 14. Visual vs verbal, as visualized in the CAM tool after dragging/selecting it from the CRT instance list; the CRT is instantiated (socket 'concepts in hierarchy' is instantiated with c1, c2, c3, c4 and c5; socket 'user input' is not instantiated) 18

Figure 15. Beginner - intermediate - advanced, as visualized in the CAM tool after dragging/selecting it from the CRT instance list; the CRT is instantiated 19

Figure 16. CAM Tool Architecture 23

Figure 17. Placeholder for a concept..... 24

Figure 18. Example of prerequisite CRT representation in CAM visual language..... 24

Figure 19. Grouping of concepts..... 24

Figure 20. Shell UML diagram..... 26

List of Acronyms and Abbreviations

Adaptive strategy	see CAM instance
Adaptive story line	see CAM instance
AHA!	Adaptive Hypermedia Architecture
AHAM	A four layered model for adaptive hypermedia
ALE	Adaptive Learning Environment
CAF	Common Adaptation Format
CAM	Conceptual Adaptation Model (sometimes used as short version of CAM model)
CAM instance	The adaptation behaviour description of a selected number of concepts, as edited in the CAM tool; an instance of the CAM model; also called: Adaptive Story Line; Adaptation Strategy
CAM model	see CAM
CAM tool	The tool (web program) that implements the requirements of the CAM model
CRT	Concept Relationship Type(s); also called: (Pedagogical) Relationship Type(s) when encountered in the CAM tool
CSRT	Concept-Service Relationship Type
DM	Domain Model; also called: Subject Matter Graph
GM	Goal (& constraints) Model
GAL	GRAPPLE Adaptation Language: the adaptation engine language
GALE	GRAPPLE Adaptation Engine
GRAPPLE	Generic Responsive Adaptive Personalized Learning Environment
GUI	Graphical User Interface
GWT	Google Web Toolkit
LAG	Layers of Adaptive Granularity: a framework for describing the power (and reuse) of an adaptation language: assembly language, programming language, procedures and strategies

LAG language	A language incorporating the middle layer of the LAG framework
LAOS	Layered WWW AHS Authoring Model and their corresponding Algebraic Operators: a five layered authoring model for adaptive hypermedia
LMS	Learning Management System
MOT	My Online Teacher
PM	Presentation Model
RM	Resource Model
SDM	Service Domain Model
SRT	Service Relationship Type
UM	User Model
WP	Work Package
XML	eXtensible Markup Language

1 Introduction

Adaptive Hypermedia (AH) is perceived to have the potential to offer a rich learning experience, personalised for each learner. To realise this potential however, authors need to have the ability to easily create adaptive material. There exist several Adaptive Hypermedia reference models, like AHAM (Adaptive Hypermedia Application Model) [36] and LAOS (Layered WWW AH Authoring Model and their corresponding Algebraic Operators) [13] that are specifically developed for authoring (instead of general purpose models of AH, such as XAHM [5], Munich [27], GAHM [32]). However, even when using tools developed based upon these models, authoring remains a difficult and time-consuming task [24]. A solution is to use graphical tools. Existing graphical authoring tools (e.g. the Graph Author developed for AHA! [17]) use concrete connections between concepts, and, the adaptivity is specified in a single layer. This approach means that the re-usability of the adaptivity is limited.

Here we present the CAM model and tool, which incorporates the overall authoring approach of the GRAPPLE project. The authoring approach in GRAPPLE is to offer a graphical tool to create a conceptual adaptation model (CAM).

The objective of the *CAM model* is to be a *system-independent high-level model for straightforward, intuitive graphical authoring* of Adaptive Hypermedia. The objective of the *CAM tool* is to be a tool that illustrates the CAM model, thus allowing for system-independent, high-level modelling for straightforward, intuitive graphical authoring of Adaptive Hypermedia.

The CAM tool is thus a tool for creating CAM instances with the help of the DM tool (as described in deliverable 3.1a) and the CRT tool (described in deliverable 3.2a). In this document relations with the DM and CRT models and tools are investigated and a design of the CAM model, CAM tool and the shell that links the three tools together is given.

The nature of the CAM model and its relation to the DM and the CRT models mean that the Conceptual Adaptation Model is formed of three sub-models, as shall be further detailed in this deliverable:

1. the graphical representation (thus graphical language) of the Conceptual Adaptation Model (the *CAM graphical language*). This language expresses how the domain concepts and CRT instances are appearing on the author's screen.
2. the generic, XML-based representation of the Conceptual Adaptation Model (the *CAM internal language*). This language is the XML-equivalent of the CAM graphical language, and describes how domain concepts and CRT instances are linked together, as well as how they are graphically represented in the author's interface.
3. the generic, XML-based export language of the Conceptual Adaptation Model (the *CAM external language*). This language describes only how domain concepts and CRT instances are linked together, without any graphical information. The CAM external language cannot (normally) be interpreted by an engine by itself, but needs to be paired with the CRT language.

The remainder of this deliverable is organised as follows. In section 1.1 we explain the structure of a CAM, based on multiple adaptation layers. Although the multi-layer model is loosely based upon lessons learnt from LAOS & LAG (Layers of Adaptation Granularity) [15] authors are not required to write "pseudo code" as they do in LAG. In section 1.2 we discuss the relations between the CAM model and tool and the Domain- and CRT- models and tools as well as the relation to the GRAPPLE engine. Section 1.3 presents related research. Section 2 extracts more CAM requirements, based on multiple scenarios of desired CAM tool use. Section 3 extracts other requirements, based on model verification considerations, and integrated authoring experience, as well as a few supplementary requirements. Section 4 proposes a design of the CAM model and tool based on the previous requirements: the CAM languages (with more details in Appendix 3). The section also proposes the GRAPPLE shell. More details on the shell are given as Appendix 1, for ease of reading of the main text. Section 5 discusses extensions to the CAM tool, for adaptive simulation and virtual reality. Section 6 concludes. A comprehensive list for the overview of all requirements is given as Appendix 2. A list of all main acronyms and abbreviations is provided at the front of the document.

1.1 The CAM model

The CAM model is based upon lessons learnt from the AHAM [13] and LAOS [13] models, as well as being inspired by the multi-model, metadata-driven approach to content adaptation [17], and has incorporated ideas from other models, such as Dexter [22], XAHM [5], Munich [27], UWE [27], ADAPT [20]. The section is organised as follows. First we will shortly introduce the (relevant parts of) the AHAM and LAOS models. Then we will investigate the specific requirements of the CAM model that are not already in AHAM or LAOS and finally we will indicate how the different layers in the CAM model interrelate.

1.1.1 AHAM

AHAM [36] is a reference model for Adaptive Hypermedia Systems (AHS) which describes adaptive applications as consisting of three main layers:

- The *Domain Model* (DM), which describes concepts, groups them in a hierarchical structure, and defines arbitrary concept relationships, possibly of a special domain *concept relationship type* (domain *CRT*). In principle a DM can be “imported” from a subject domain ontology, except for the concept relationships that have a (pedagogical) meaning.
- The *User Model* (UM), which also defines concepts, but with user specific attributes, e.g., knowledge level, learning style, preferences etc. Typically the UM is an overlay model of the DM, meaning that for every concept in DM there is a corresponding concept in the UM.
- The *Adaptation Model* (AM), which defines the adaptation behaviour. It typically consists of condition-action rules or event-condition-action rules that define how user actions are transformed into UM updates and into the generation of presentation specifications. There are two types of rules:
 - *generic adaptation rules* are connected to CRTs; this for instance allows to define a knowledge update rule for visiting pages and a prerequisite rule for determining the suitability of concepts depending on whether all prerequisites are satisfied; an author only has to specify concept relationships and an authoring tool can then generate the corresponding adaptation rules automatically;
 - *specific adaptation rules* apply to specific concepts of the domain model and can be used for a very rare adaptation rule or for defining an exception to a generic adaptation rule; authoring such specific adaptation rules requires knowledge of the language in which such rules are defined (and which is system-dependent).

In the AHA! system [24] a graphical authoring tool, the Graph Author, is used to define the DM and to draw a graph of concept relationships (of different types). As the name “graph” already suggests concept relationships are (unary or) binary, whereas in AHAM there is no restriction to the number of concepts that together may form a relationship. This Graph Author is the inspiration for the CAM tool behaviour (in terms of ‘look and feel’). However, the functionality of the CAM tool is highly enhanced, as we shall see.

1.1.2 LAOS

The LAOS model [13] is an extension of AHAM. The AM in AHAM has rules for updating the user model (e.g., with knowledge values), for defining aspects of the presentation (e.g. the presentation style for links depending on their suitability) and for domain-independent but only user-dependent aspects (e.g. a learning style). In LAOS, different aspects of the adaptation model are distributed over multiple layers in the model [31], in particular the:

- *goal and constraints model* (GM), extracting and concentrating all pedagogical information previously intermingled with domain information in the AHAM model;
- the *adaptation model* (AM), extending the ideas of general-purposeness and specificity as follows:
 - *generic adaptation rules*: instead of being connected to an event type (such as visit), it allows for specification of the combination of *type of event*, *type of concept*, *type of relation* and determining, based on this ternary combination, the UM or PM updates. Such rules can be applied to *any* domain map.
 - *specific adaptation rules*: similar to the generic rules, but applied to a specific concept, identified from a given domain map.

and

- the *presentation model* (PM), extracting and concentrating all presentation information previously intermingled with domain information in the AHAM model.

In this way, pedagogical information can be expressed in the GM, and kept apart from other information. Also, the PM describes the final look & feel of the presentation as well as quality of service parameters (e.g. for mobile devices). CRT's in LAOS are also of different type, depending on the model they belong to: e.g.,

- the domain CRT's only describe domain relations (such as part-of, IS-A or relatedness relations as available in the MOT authoring tool [8] built on LAOS); similarly,
- CRT's in the GM only describe pedagogic relations (such as AND-OR relations with pedagogical labels, as available in MOT).

Moreover, the adaptation model allows for different levels (and thus degrees) of reuse of adaptation, by conforming to the LAG framework [4], [10]: establishing as a first level the (event-)-condition-action rules, as in AHAM, that are the building stones for adaptation, and also thus *assembly-language type of adaptation* specifications; at the second level, allowing for more sophisticated *adaptation languages* [8] (such as LAG [10] [15] or LAG-XLS [12], [33]) ; finally, at the last level, *adaptation strategies*, comprising reusable, annotated storylines of adaptation and personalization, that can be applied to various domain models. The higher level of separation of concerns in LAOS, as well as the LAG language is the inspiration for the CAM model, and the CAM languages, respectively. However, the languages are targeted at different authoring aspects and capabilities, in the spirit of the LAG model, but extending this model considerably.

1.1.3 CAM

In the GRAPPLE project, the authoring framework used is even more general and flexible: it contains an extensible number of layers, which may be different for each application. This is due to the fact that, for instance, for a specific application, page-presentation and quality-of-service parameters might need to be stored independently, if they are major components of that application, whilst for another one, the approach of storing them together as in LAOS is acceptable. Thus, application-dependent new layers should be allowed to be defined by authors. However the DM, UM and AM layers are described as mandatory, as without these basic functionalities no adaptive system will function (DM for the underlying domain information, UM to describe the user and the AM to describe the adaptation rules).

This leads to the following list of *General Requirements* for the GRAPPLE toolset:

GR1. The GRAPPLE authoring tool should allow an extensible number of layers, however a DM, UM and AM layer are mandatory. (as these are present in both AHAM and LAOS)

GR2. The GRAPPLE authoring tool should be visual. (this is the major requirement from GRAPPLE project specifications)

GR3. The GRAPPLE tool should contain a UM tool/service (matching the mandatory UM layer of GR1).

GR4. The GRAPPLE authoring tool should contain a DM authoring tool/service (matching the mandatory DM layer of GR1).

GR5. The authoring tool should contain an AM authoring tool (matching the mandatory AM layer of GR1).

The combination of GR2 and GR5, as well as the prior experience with the AHA! graphical authoring tool led to the conclusion that adaptation authoring should deal separately with CRT types and their application. Thus, GR5 is subdivided into:

GR5.1: The GRAPPLE authoring tool should contain a CRT type tool.

GR5.2: The GRAPPLE authoring tool should contain an adaptation strategy (or story line) creation tool, called CAM tool (conceptual adaptation model tool).

There will always be a DM and UM layer and at least one layer with adaptation aspects, so the structure of GRAPPLE authoring is a generalization of the AHAM model, and either equivalent with, or a generalization of the more refined LAOS model. Thus, the GRAPPLE authoring shell tool will comprise a DM authoring tool, UM authoring tool, a CRT authoring tool and a CAM authoring tool. The presence of the CRT ensures that the complexity of authoring specification is hidden in the definition of these concept relationships, whilst the

presence of the CAM tool ensures a high-level, non-programming access to authoring behaviour specification. In this deliverable, due to its important role in integrating all the other authoring elements, the focus is on the *CAM model and tool*. Deliverable D3.1a deals with the DM tool, and deliverable D3.2a deals with the CRT tool respectively.

Because of the limited nature and specificity of the DM and CRT authoring, and of the general requirement GR1, the following requirements (R) for the CAM authoring tool¹ can be expressed:

R01. The CAM model and the CAM tool built on it should allow for an extensible number of layers.

R02. DM concepts and relations representation should keep the same look and feel between different layers (DM, CAM layers).

R03. CRT relationships and placeholders should keep the same look and feel between the different layers (CRT and CAM layers).

As the CAM tool inherits the requirements from GR2, as well as is influenced by R02, R03, the requirement R04 can be defined as follows:

R04. The CAM tool should be able to visually represent the different layers (e.g., showing only one type of relationship or showing multiple relationships, but each with a different representation - in the simplest case, colour).

The relationships defined in the different CAM layers do not yet express the actual adaptation that will take place. A prerequisite may be translated to a rule that will change the presentation of links to concepts, but it may also be translated to the conditional inclusion of a prerequisite explanation (fragment). The translation of CAM instances to actual adaptation rules is described and exemplified in Section 3, 4.

1.2 Relations between CRT tool and other relevant tools of the GRAPPLE project

1.2.1 Relation to the DM tool

The Domain Model (DM) describes the domain concepts, their attributes and their semantic relationships. These relationships do not attach any adaptive behaviour but merely indicate a semantic link. In the CAM tool, concepts from the DM tool can be selected and the adaptive behaviour can be attached to these concepts. For a concept to appear in the final adaptive lesson, it needs thus to be selected in the CAM tool. This implies the following requirement:

R05. There should be a common shell for the DM and the CAM tools, and a drag & drop facility from the former to the latter, to allow for one (or more) domain concepts to be selected and dragged into the CAM tool.

Figures 1 and 2 show the interaction of the CAM tool with the DM tool.

1.2.2 Relation to the CRT tool

The CAM model only puts together the final picture of the adaptive behaviour, much like the pieces in a jigsaw. The types of possible adaptive behaviour are defined in the CRT library created by the CRT tool. The CAM tool can link a number of domain concepts using a certain CRT, and therefore concretely establish the adaptive behaviour defined in the CRT in a more generic way. CRTs dragged into the CAM tool can be then populated with domain concepts from the DM tool, as long as the particular CRT allows for that particular type of concept.

Figures 1 and 2 show the interaction of the CAM tool with the CRT tool.

1.2.3 Relation to GRAPPLE Authoring Shell

The CAM tool functions within the GRAPPLE Authoring Shell that also embeds the DM tool and the CRT tool.

Note: the Google Web Toolkit (GWT) was proposed for this common shell, after a lengthy process that involved all the partners. The main reasons for its recommendation were the fact that it's free for all partners,

¹ Please note that the (requirements on the) UM is (are) specified implicitly in CRT's and the CAM. The GRAPPLE project envisions a distributed user modeling service that will take these requirements and transform them into a, potentially distributed, user model.

based on a language that everyone knows at least to some extent, has certain visual provisions and has the provisions specifically required for VR and simulations.

1.2.4 Relation to Adaptation Engine

The CAM model groups domain descriptions and adaptation descriptions into adaptation strategies or adaptive story lines. These story lines need to be exported to an adaptation engine. This engine can be the GRAPPLE adaptation engine developed in the GRAPPLE project or any other engine that can work with or has converters for the exported adaptation language. The CAM tool will export the *CAM external language*, a combination of DM and CRT-instances. Figures 1 and 2 show the interaction of the CAM tool with the Adaptation Engine.

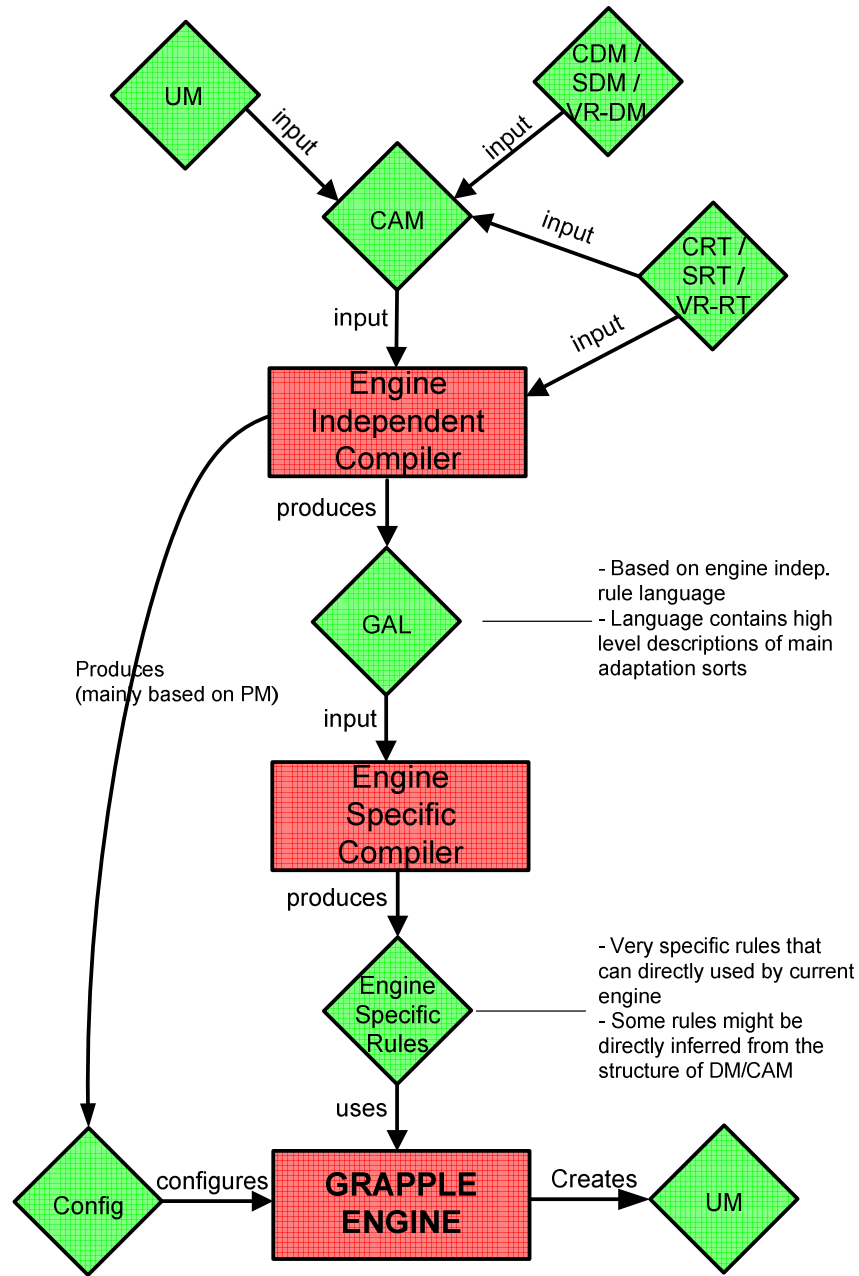


Figure 1: Information flow of GALE (Grapple Adaptation Engine) components and its input from the CAM (and related models)

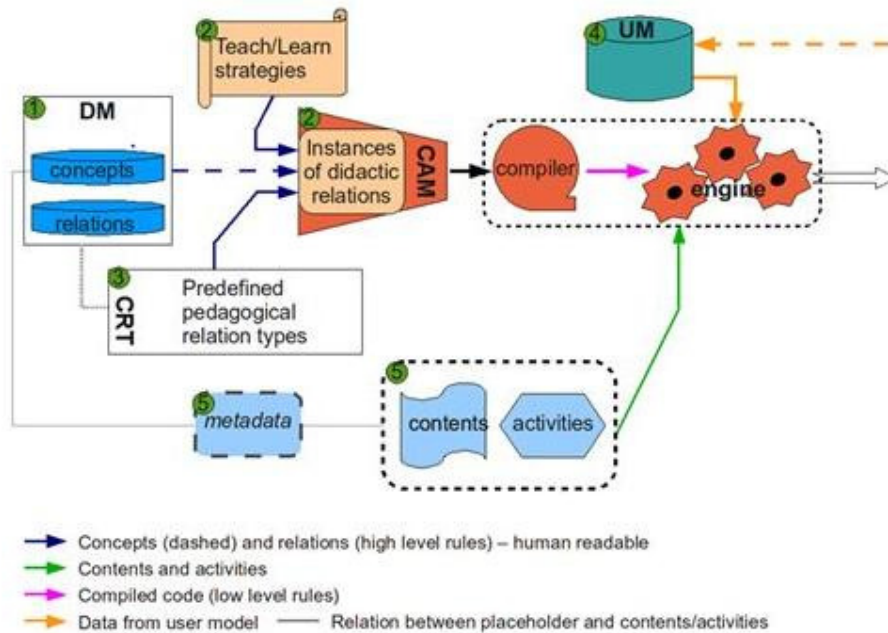


Figure 2. GRAPPLE Software Architecture (taken from deliverable D3.1a)

1.3 Related work

Authoring of adaptive hypermedia is notoriously difficult work [3]. Research on improving this process ranges from ontology-based authoring [28], to integrating standards and their representations [21], [25], using data mining techniques [35], web services [29], interfacing techniques between authoring systems [9] and adaptation languages [10].

The current work is based on prior developments of adaptive hypermedia frameworks, like AHAM [36] and authoring frameworks for adaptive hypermedia, such as LAOS [13] and LAG [10]. Moreover, it is based on systems for adaptive hypermedia delivery, such as AHA! [17] and for authoring of adaptation, such as MOT [14], My Online Teacher [8], APels [18], ACCT [16].

Any Adaptive Hypermedia authoring system has to choose a suitable representation of the Adaptive Courses. While previews are certainly required, editing in previews will be too confusing. As we can see in [19], the two main candidates are tree-based and graph-based representations. Experiments show that graph-based representations are a little more difficult to learn, but are better suited to navigation and orientation in Adaptive Hypermedia structures than tree-based ones.

Finally, this research is based on evaluations of authoring processes for adaptive hypermedia, as performed with various groups of students, in various locations, and with different versions of constantly improving tools, including but not limited to: [14], [6], [11], [9], [23], [16], [7]. Such research shows that, whilst having a higher flexibility and multiple layers for authoring is advantageous [7], [10] it is difficult for authors to actually program the adaptive behaviour of adaptation strategies [11], and it's thus much easier to have them reuse strategies at a higher granularity level, in a graphical interface [7]. As the best paper of the 4th International Workshop on Authoring of Adaptive and Adaptable Educational Hypermedia (A3H) shows [30], a template-based approach of a graphical nature is easier to handle for teachers, who in this way can better make use of the flexibility that the CAM GRAPPLE tool is offering.

2 Scenarios of CAM (tool) use

2.1 A generic illustrative scenario

We illustrate the authoring process of a CAM instance by means of a scenario in which a teacher needs to express some generic and specific prerequisite relationships.

Dr. Brown² prepares a new on-line course on History of Art for first year undergraduate students. He has two options: he can either try to define a link structure between the course pages in such a way that students never see a link to information they cannot yet understand (because of missing foreknowledge) or he can define a CAM instance with prerequisite relationships and then rely on the adaptive learning environment (ALE) to ensure that students are only guided towards pages for which they have all prerequisite knowledge. Although it is often argued that defining adaptation (a CAM instance in this case) means creating an adaptive course and is more work than creating a static course, the converse is actually true: the first option, to create a static course that is such that students can only follow links to information they are ready to understand is a nearly (or perhaps completely) impossible task and would require a lot of very careful work in selecting links to show to (all) students.

Requirements arising from this:

Environment:

RE1. The CAM tool should allow for graphical, drag and drop input.

Functionality:

Interfacing:

RI1. The CAM tool should interface with the DM tool to select concepts (or groups of concepts).

RI2. The CAM tool should interface with the CRT tool to select CRTs (or groups of CRTs).

Verification:

Dr. Brown realises that there are many ways to present the material of his course. He decides that he first wants to introduce the artist, then teach about the artworks and then teach about the bigger picture, i.e., about the style and period the artist belongs to. He however decides he does not mind which artist students start with. (Note that Dr. Brown could have chosen to present first the artworks and then their creators, or even an entirely different strategy altogether.)

Dr. Brown wants to create a prerequisite relationship from “Michelangelo” to “The Last Judgment”, as the students should first learn something about the artist before learning about the artist’s artworks, and then do this for every artist in his course, to end up with the desired result. The authoring tool allows authors to draw a prerequisite relationship between a set of (prerequisite) concepts on the left and a set of concepts on the right. First he drags the prerequisite relationship on the canvas, this would look as follows:

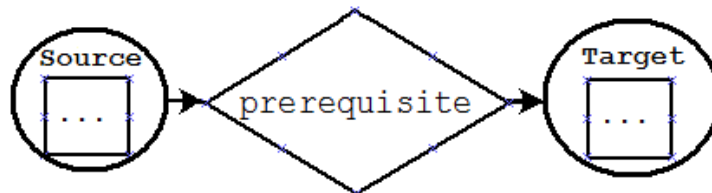


Figure 3 A CRT in the CAM tool: here, a prerequisite relation, with empty sockets (placeholders) for concepts. Then he drags the concepts *Michelangelo* and *The last Judgment* in the sockets source and target, respectively. After that the display would look like:

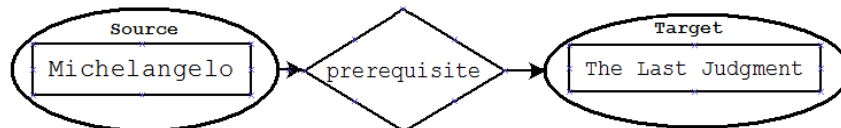


Figure 4. An instantiated CRT in the CAM tool: here, the Relation between Michelangelo and The Last Judgment

This would create the result Dr Brown ultimately wanted to achieve. However, Dr. Brown realizes that “Michelangelo” is not just a prerequisite for “The Last Judgment” but for every artwork by Michelangelo. He

² Any resemblance with an existing person is purely accidental.

realizes he can have the amount of work he has to do reduce and decides thus to use a different CRT instance:

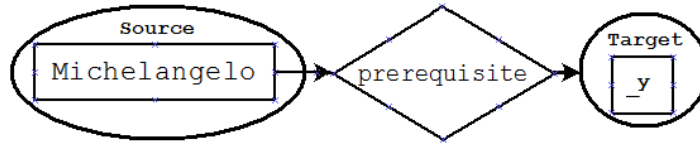


Figure 5. Relation between Michelangelo and Placeholder Concept `_X`

This new CRT is also a prerequisite with the condition:

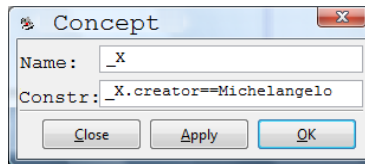


Figure 6. The placeholder represents all concepts for which the creator is Michelangelo

The specific concept relationship is replaced with a partially generic one: there is still one specifically named concept but also a variable to express that the relationship applies to all concepts `_x` that satisfy a certain condition. The underscore indicates that `x` is a variable and not a literal value.

Something perhaps not immediately obvious from this example is that there are two possible uses of this authoring tool (plus a combined third one):

- In the example, the “creator” attribute is a DM property, probably derived from a subject ontology. Which concepts have “Michelangelo” as prerequisite depends purely on the DM and this is thus independent of the learner taking the course.
- It is equally possible to use an attribute from the UM in a relationship, thus creating relationships that are not only user-dependent but dependent on the “current” instance of the user model.
- There is finally a third case, by combining the previous two. The learning application can for instance *recommend* topics from a list that first of all depends on the DM but that also depends on the user’s knowledge. For instance, only those recommended topics may be shown of which the user still has little or no knowledge.

Note that when the relationship only depends on DM information (like in the example) the replacement of `_x` by actual concepts could (but need not) be done at compile time, i.e. when translating the CAM instance into actual low level adaptation rules to be executed by the GRAPPLE ALE. When the relationship depends on UM information this is not possible.

Dr. Brown may later also go one step further in the definition of the prerequisite relationships. He may wish to state that for every artist and artwork the learner should learn about the artist before studying the artworks from that artist.

The CRT he will then use instead is something like:

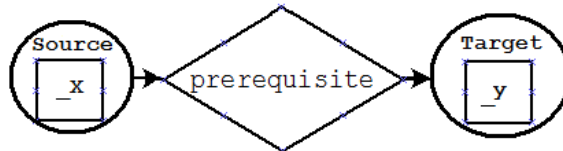


Figure 7. Another non-instantiated CRT in the CAM tool: Relationship (CRT) for generalization of the Michelangelo example

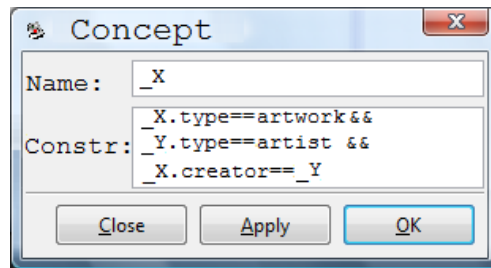


Figure 8. Constraints for generalization of the Michelangelo example

Note that whereas creating a (set of) *specific* concept relationships does not require any knowledge of the structure of the DM or UM or any language to refer to DM or UM attributes of concepts, creating *generic* concept relationships, or CRTs does require some basic knowledge of the CRT language (to write `_X.creator==Michelangelo`).

Dr. Brown may not have to define the CRT himself. He would just use an appropriate one from the CRT library. This is why the descriptions of the CRTs are very important for Dr. Brown to make sense on how to use CRTs written by other authors or programmers. More about the creation of CRTs used by the CAM tool can be found in the deliverable D3.2a.

An alternative way to define the actual adaptive behaviour associated with a relationship is to just define a *method call* for a method that needs to be defined in the translation model. This approach makes the use of the CAM model (and CAM instances) very powerful and generic but it also makes the behaviour dependent on a low level implementation rather than a high level specification. It is unlikely that teachers (like the imaginary Dr. Brown) will resort to writing program code for the adaptation engine.

2.2 Pedagogical strategies in the CAM tool

In the previous section we have seen a scenario illustrating how a teacher can create or customize an adaptive lesson. Previous research has defined interesting pedagogically sound adaptation strategies, representing different learning scenarios based on learners' needs, preferences, some also based on complex (and controversial) pedagogical foundations, such as learning styles, for Adaptive Hypermedia³ [1]. In this section we will explore some of these strategies in relation to the CAM tool. More specifically we will check how, in principle, such strategies can be expressed in the new CAM tool and CAM languages. As the CAM model is aimed to be richer than previous attempts, it should at least be able to express the basic strategies we have defined before via languages such as LAG [15] and LAG-XLS [33]. The CAM model and the CAM tool is more flexible, however, and can express strategies beyond what is analyzed here.

Please note that the strategies chosen here are a selection, based on previous experience, and the aim is not to provide a complete set of strategies, but merely to try to understand how existing strategies could be expressed in the new model. We have therefore selected some of the technologically more challenging strategies and omitted others perhaps more interesting from a pedagogical point of view.

Rollout

The rollout strategy is a very simple strategy that allows authors to decide when a certain concept or concept part should be shown: concepts to be shown after a certain number of steps could be classified as '*showafter*', and attached the meta-data containing the number of steps after which to be shown. Similarly, concepts classified as '*showatmost*' should only be displayed at most the given number of steps as again contained in meta-data. Pedagogically, such a strategy is useful when, e.g., beginner users need more information at the beginning, which is irrelevant later on and needs to be removed (thus using '*showatmost*' for a certain number of steps). Similarly, not all information should be made available from the start, to avoid cognitive overload. Some of it can appear later, when the learner is ready for it (thus using '*showafter*' a given number of steps). The roll-out strategy depends upon the tree hierarchy. We note that it is straightforward to create such a hierarchy with the introduction of a parent-child relation.

³ See also our strategies page: <http://prolearn.dcs.warwick.ac.uk/strategies.html>

First, authors need to be able to sort the concepts in the desired hierarchy (if this is not already available, e.g., if concepts are grouped in a graph). Next, we discuss the representation of the 'showafter' part. The strategy demands that a concept is shown after its parent has been viewed a given number of times. As a constraint on `_X`, we have the following:

```
_X.metadata == 'showafter' && _X.parent ==_Y && UM._Y.showcount >=
_X.showafter
```

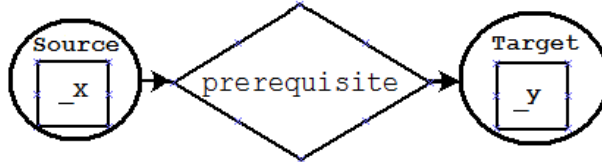


Figure 9. 'Showafter' relationship; as visualized in the CAM tool after dragging/selecting it from the CRT instance list; the CRT is not yet instantiated (no DM concepts appear in the Source and Target sockets)

In Figure 9, the relationship for 'showafter' is created via a prerequisite. This uses the prerequisite relation in its sense of condition on displaying concept `_x` based on viewing concept `_y` (and some supplementary conditions, as above). However, this does not use prerequisite in terms of knowledge update.

Depending on the implementation of prerequisite relationship, the 'showatmost' part may or may not be needed. If the implementation of the prerequisite relationship makes sure that concepts for which previously the prerequisite was fulfilled, but for which this is no longer the case, are hidden, we do not need to do anything for the 'showatmost' part. If this is not handled by the prerequisite we have to add a relationship that hides concepts once they have passed their 'showatmost' threshold.

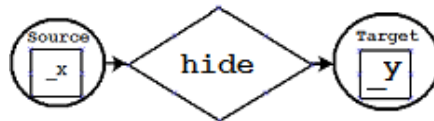


Figure 10. 'Showatmost' via hide relation (only needed if prerequisite does not hide concepts); as visualized in the CAM tool after dragging/selecting it from the CRT instance list; the CRT is not yet instantiated (no DM concepts appear in the Source and Target sockets)

The constraint is then:

```
_X.metadata == 'showatmost' && _X.parent ==_Y && UM._Y.showcount >
X.showatmost
```

Note that we also need to make sure that for each concept a count is kept in the user model. This can be done with a relationship 'countaccess' relating a concept to itself.

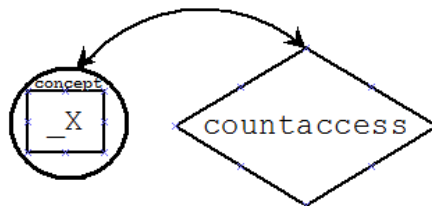


Figure 11. 'Countaccess' relationship; as visualized in the CAM tool after dragging/selecting it from the CRT instance list; the CRT is not yet instantiated (no DM concepts appear in the Source and Target sockets)

The constraint will then be:

```
_X.access == true
```

The implementation of the 'countaccess' relationship simply increases the count:

```
UM._X.showcount = UM._X.showcount+1
```

Here, it is also important to consider where the metadata is added, classifying `_X` as 'showatmost' (or 'showafter'). Unlike in the previous example (Figure 8), where the type of the concept as 'artwork' was expected to be known from the domain model, it is not acceptable that this label will be added in the DM, due to its pedagogic and adaptation-specific nature. Thus, two possible processing methods for this exist:

1. the label (metadata) is not added; simply by dragging & dropping the concept from the domain map into the right side of the placeholder CRT in Figure 9, the CAM tool will deduce that the domain concept will be labelled 'showafter'.
2. a separate labelling step for adding pedagogical or adaptation-specific metadata to domain concepts is allowed – much as in the Goal Model (GM) in the LAOS framework. The processing would be similar to labelling concepts in the domain model, but it would conceptually be happening in a pedagogical space. Then, the CRT can be applied to the whole goal model with the pedagogical labels. The advantage of this second method is that the hierarchical structuring can also be performed, if necessary (i.e., if the original domain map is a graph).

Requirements arising from this:

R06. circular CRT relations should be allowed.

Depth First

The depth first strategy is used for sequential learners. One topic at a time is presented, and the student is allowed to go in-depth (hence, the name) in this topic first, before he proceeds with the next topic. Preferably, no menus are shown to such students, and all they need to access is a 'next' button, taking them to their next study material, whether statically linked, or adaptively generated.

For the depth first strategy, again, the concepts have to be ordered in a hierarchy first. The socket that can contain any number of concepts is suggestively called *concepts in hierarchy*, but this is actually an assumption when using this relationship and is NOT enforced by the socket in any way. After this, a few relations are needed. Thus, we introduce a relation from each concept to each of its children, called *Show next child XOR next sibling*, see Figure 12.

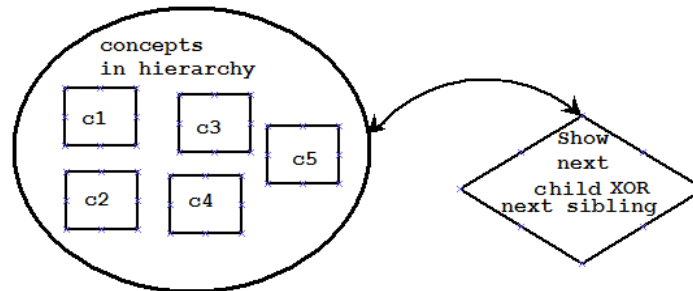


Figure 12. The main relation implementing the 'Depth First', the logic in the constraint takes care of showing the appropriate next concept, either the next child or the next sibling; , as visualized in the CAM tool after dragging/selecting it from the CRT instance list; the CRT is instantiated (the socket contains concepts c1, c2, c3, c4 and c5)

The condition must ensure that $_x$ is the next sibling of $_y$ that needs to be shown, as well as update the User Model variable that keeps track of the current position of the learner within the hierarchical course. The condition shall only show the next sibling if the concept does not have any children left to be shown.

Finally we add the root domain concept, which shows first the concept unconditionally.

This is a good example where, in principle, a domain relation can be translated directly into a CRT. Thus, the CAM tool should be able to specify that, for a given DM tree, each *parent*->*first-child* relation, as well as each *finished-child*, *sibling*->*next-sibling*, as well as each *finished-child*, *last-sibling*->*next-parent* relations should all be transformed into prerequisite relations.

Requirements arising from this:

R07. Conversion of domain relations into CRTs should be possible in the CAM tool.

R08. Pedagogical labelling should be possible in the CAM tool.

Breadth First

The breadth first strategy is used for global or holist learners. These learners like to see the global 'picture' first, before they dive into any topic. For such students, menus and other orientation devices are quite helpful.

Thus, implementation of this strategy has to start with the ordering of the concepts in a hierarchy. Next, we draw relations between each concept and each of its children, allowing them to show (all) the children if the parent has been shown. Finally we create a relationship from the root to the root, which shows the first concept unconditionally.

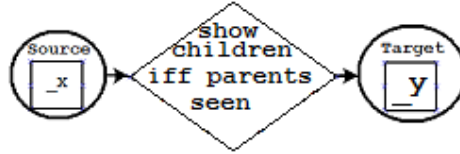


Figure 13. The relation shows `_Y` if `_X` has been shown the condition is: `_Y.parent==_X`, as visualized in the CAM tool after dragging/selecting it from the CRT instance list; the CRT is not yet instantiated (no DM concepts appear in the Source and Target sockets)

Visual – Verbal

Visual-verbal preference corresponds to a strategy which does not need concept ordering. Students are shown visual material (graphs, pictures, video, flash, simulations) if they have a visual preference, and verbal material (text, audio, etc.) if they have a verbal nature. For visual-verbal we need only one relationship, one standard prerequisite relationship with a constraint like `_X.label == UM.preference`. We also need to be able to design some menu where the learner can set his preferences, or to introduce a number of settings concepts which are always visible and a relationship on those, which manipulates the preference in the users UM. Alternatively, the UM variables can be set via an initial questionnaire, or test.

Here, beside the issue of the place in the authoring where the labels are set (similarly to the Rollout discussion), the issue of creating a CRT which specifies as behaviour user input appears. In some existing Adaptive Hypermedia systems a work-around with special concepts are used. Accessing these triggers the desired setting. At this point it has not yet been decided whether a dedicated user-interface mechanism will be developed, however the work-around will also be possible in a CAM instance.

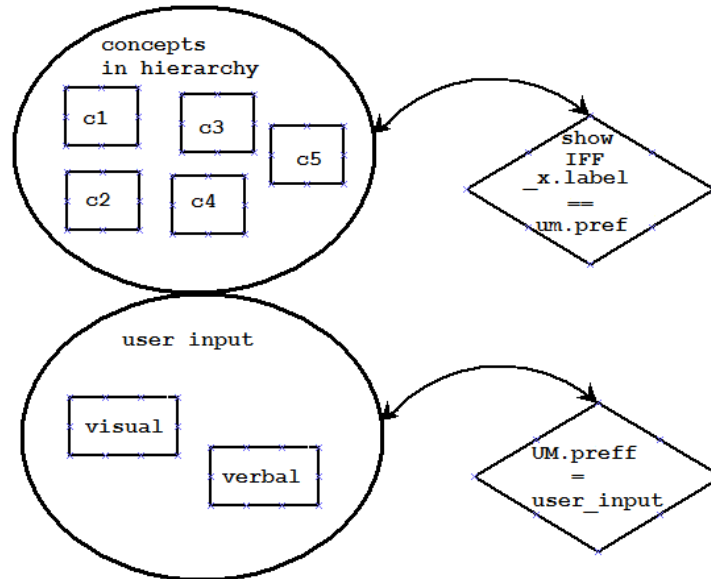


Figure 14. Visual vs. verbal, as visualized in the CAM tool after dragging/selecting it from the CRT instance list; the CRT is instantiated (socket 'concepts in hierarchy' is instantiated with c1, c2, c3, c4 and c5; socket 'user input' is not instantiated)

Beginner – Intermediate – Advanced

The beginner-intermediate-advanced strategy is a fundamental strategy, dividing the available course material into three types: material aimed at beginners, material for intermediate students, and material for

advanced students. Typically, students that have been identified as beginners are not allowed to see material from the higher levels till their status is changed (by, e.g., reading all material in their level, or taking a test).

For the strategy implementation in the CAM tool, we drag a relationship called *beginner intermediate advanced* onto the canvas. This relationship has three sockets that each can contain an arbitrary number of concepts and are suggestively named beginner, intermediate and advanced. The relationship shows all concepts in the beginner socket, hides all concepts in the intermediate and advanced sockets first. It then shows the concepts in the intermediate sockets if all concepts in the beginner socket have been seen and shows all concepts in the advanced socket if all concepts in the intermediate socket have been seen. Please note, this requires the sets of concepts to be strictly disjunctive. I.e. a concept can not be in the beginner and intermediate socket at the same time, as this would hide part one of the beginner concepts and create a deadlock. An alternative would be to specify additional conditions for the concepts, such as the fact that only resources with specific labels are allowed in a given socket. For instance, only resources in image format are allowed in the beginner socket, and text format for intermediate socket, etc. In this way, the same concepts could be dragged and dropped into different sockets, meaning that their different representations (corresponding to their different labels) can be shown when the different sockets are triggered.

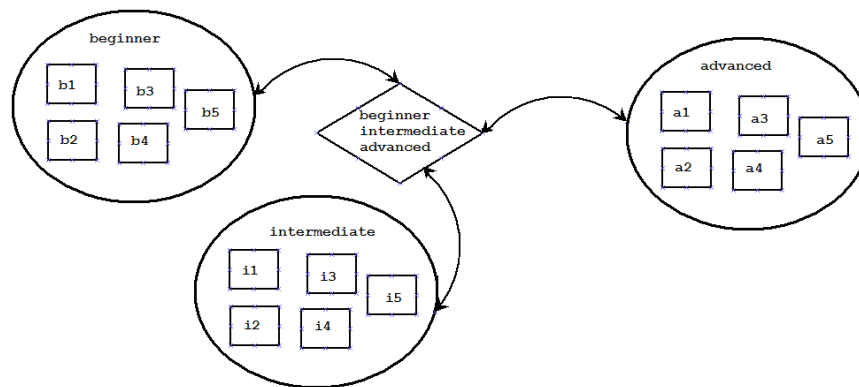


Figure 15. Beginner - intermediate - advanced, as visualized in the CAM tool after dragging/selecting it from the CRT instance list; the CRT is instantiated

Relatedness

The domain model (DM) used by the CAM tool can inherit multiple layers of relations. These relations can be further used in the adaptation process. For instance, an advanced student can be shown all related concepts, whereas a beginner student is only shown concepts within his own course.

For the relatedness strategy we draw a relationship between `_Y` and `_X` that shows `_Y`. We then use a condition like: `_Y IN _X.relatedness`

In this section we have analyzed various adaptation and pedagogical strategies, and how the CAM tool would be able to represent them and improve the access of teachers to such complex adaptation notions. In such a visual way, teachers can get to grips with the authoring tasks more easily. Next, we are going to analyze issues appearing in the authoring process, which need to be handled transparently, by the authoring system, without the direct knowledge of the authors.

Conclusions

While trying to express the (selection of) learning style related strategies we noticed some common issues:

- It is clear that we need to have some view of the Domain Model in order for the teacher to see what the available concepts are.
- A wizard like interface for ready-made strategies could be very helpful, while still allowing customization.
- The step-wise processing as previously implicitly assumed in LAOS/LAG based systems is still desirable. Otherwise some strategies like the Breadth- and Depth-First will not be possible, as inference rules will make sure the whole content will directly be visible. Thus, rules need to be

triggered one-step-at-a-time, when certain events occur (e.g., a mouse-click). It is envisioned that, if desired, it should be possible to specify rules that trigger other rules, like in AHA!, however, in a visual way.

- In the LAOS/ LAG conversions to AHA!, one could control to a certain extent what kind of menus and other guidance the student would get. This represents adaptation of the presentation layer in LAOS, and reflects on interface changes and display for the student. It is desirable that in the new CAM-based systems this control will also be present to some extent.

3 Other requirements

3.1 Model verification

The authoring process (for the concept structures and the adaptation) which is focused on the creation of concept relationships, appears to be fairly simple. Using different layers for different *CRTs* makes understanding the conceptual structure relatively easy too. However, this simplicity is partly an illusion. Depending on how concept relationships are translated (using a translation model) to the low level adaptation rules for the adaptation engine, the (graph-like) structure of concept relationships of a single layer may already cause problems, and the combination of concept relationships from different layers may cause even more problems. We illustrate this with some examples.

Consider a simple structure where A is a prerequisite for B, B is a prerequisite for C and C is a prerequisite for A. This may cause a problem or not, depending on how prerequisites are used in the learning application.

- When “A is a prerequisite for B” results in links to B being recommended only after learning enough about A; it is possible that the cycle of prerequisites causes the links to A, B and C to never become recommended to the learner. (Needless to say this is a problem.)
- When “A is a prerequisite for B” means that a short explanation of A will automatically be inserted into a page about B to compensate for the missing foreknowledge then there need not be a problem. If A is accessed first it will contain a prerequisite explanation of C, possibly preceded by a prerequisite explanation of B. (In this way the cycle does not cause a problem.)

Problems with undesirable structures like cycles are relatively easy to detect within a single layer. The problems become much more unpredictable when looking at the adaptation rules that result from translating the concept relationships from all layers together. The most common types of problems are *termination* and *confluence*.

3.1.1 Termination Problems

A simple example of where rule execution can run out of hand is when an author creates *knowledge propagation* relationships. A page that is essentially about Michelangelo may contain a brief description of some of his masterpieces, like “The Last Judgment”. Our imaginary Dr. Davies may draw a “10% knowledge propagation” relationship from “Michelangelo” to “The Last Judgment”. However, there may also be a generic rule that states that whenever you learn something about an artwork you also learn something (maybe also 10%) about the “creator” (artist) of that artwork. It is possible that the *knowledge propagation CRT* has a translation model that will cause the translation of such a cycle to be an infinite loop of rule executions. (Each knowledge increase of “Michelangelo” may involve a knowledge increase of “The Last Judgment” and vice versa.) Disallowing cycles within a layer guarantees that there are no *termination problems* within that layer. However, even when each layer is without termination problems the interaction between rules of different layers may still cause an infinite loop.

The *static analysis* proposed in [36] results in conditions that may be too restrictive to apply them in multi-layer CAM instances. The authoring tool might well disallow the creation of harmless concept relationships just because the static analysis detects a cycle, even when no infinite loop would be possible (when actually considering the conditions of the rules and the possible effect of the actions of the rules).

So rather than performing such static analysis, it is possible to apply a heuristic that is applied at runtime (in the adaptation engine) and that will ensure that there are no *termination problems*:

- The first step is to perform static analysis to ensure that no termination problem can be caused by the rules associated with the relationships of any single layer.
- The second step towards a solution for termination is to assign a (different) *priority* to each layer. (This is not to be confused with *execution phases* of AHAM [36]. This is similar to priorities for adaptation strategies in the LAG language [10], [15].)

- The third step is to disallow updates to an attribute A of a concept C when C.A has been updated already by a rule associated with a higher priority layer or when an update to C.A already triggered the execution of a rule at a higher priority layer. (Note that just ensuring C.A has not been updated by a rule of a higher level is not enough. The C.A updates as a trigger is really a necessary additional condition.)
- For updates in the same layer, with the same priority, a warning should be given to the author that the same attribute C.A is updated in various places and needs confirmed.

Although this method ensures that infinite loops are not possible, it makes the behaviour of the adaptation engine dependent on the choice of the priorities of the layers. We expect such problems to be rare, but nonetheless a system designer should determine the proper priorities for the “predefined” layers that are made available to authors (who do not define their own CRTs and translation models).

Requirements arising from this:

R09. The CAM tool should allow for simple static analysis for possible termination problems, as far as possible without evaluating the GAL code in CRTs.

3.1.2 Confluence Problems

Confluence problems occur when more than one rule tries to update the same attribute of the same concept. The order in which such updates are performed may determine the resulting UM state.

- Static analysis can be used to ensure that there are no confluence problems within a single layer.
- In addition to this analysis we again assign a (different) *priority* to each layer and we disallow updates to attributes of concepts that were already updated at a higher (priority) level.

Like for termination, the assignment of priorities to layers may potentially influence the outcome (the UM instance) of the adaptation rule execution.

Requirements arising from this:

R10. The CAM tool should allow for priorities to be set to the various layers.

3.1.3 CRT and DM combination Problems

If any concepts are allowed for any CRT relations, it is possible that the adaptation may not work properly. We are envisioning the following issues, and allow for the corresponding corrective actions, as follows:

- **Wrong arity** in CRT: For instance, if a certain prerequisite needs only one concept on the left side, and one on the right, but multiple are dragged into the left (or right) side, this should trigger an error.
- **Wrong type** in CRT: Similarly, if a certain type of metadata is expected in the placeholder of a CRT (such as `_Y.type=artist` in Fig. 5) and the concept dragged is not of that particular type, an error should be triggered and the drag & drop should not be allowed.
- **Wrong CRT combination:** for instance, when two perfectly acceptable CRTs are combined, but when the result is not acceptable (e.g., when the right side of the first CRT is not compatible with the left side of the following CRT). In this case, again, an error should be triggered and the combination should not be allowed. If the combination is possible, but results in an enhanced set of constraints, the DM concepts dragged into the overlapping placeholder would have to respect both sets of constraints.

Requirements arising from this:

R11. The CAM tool should allow for checking of combination problems. (This also implies that such information is available to the CAM tool from the DM and CRT tools respectively).

3.1.4 Conclusions on Model verification

Authoring adaptive behaviour using graphs makes the authoring process easier, but also increases the change of authors introducing potential problems e.g. termination and confluence. It is unfeasible to do a full verification of the graphs in the CAM tool. The reason behind this is that the graphs are conditional, so static structure analysis alone will not be enough. In general, verification would imply simulation all possible paths. In the scope of the CAM tool project it has been decided to do a partial verification. CRT's will define how

many and what type of entities need to be connected to it. The CAM tool will verify this when the author tries to make the connections. It will also warn the author about cycle structures in the graphs of the CAM instances, but will not prevent these as conditions in the CRT's may mean that in practise there is no problem (no execution paths containing a cycle exist).

3.2 Other requirements

The CAM tool needs to be as intuitive as possible. A known problem with adaptive environments is that authors cannot get the WYSIWYG (what-you-see-is-what-you-get) functionality. However, different simple previews are possible, such as showing an author all content selected in the order that it was selected, or even simple instantiations of user model variables (showing the author what a beginner student would see, for instance). Thus, preview of the authored CAM instance is necessary.

A full-scale adaptation is provided by the adaptation engine, so this should not be duplicated as such. However, a small amount of pre-adaptation should be visualized via the CAM tool.

Requirement arising from this:

R12. The CAM tool should allow for a preview button

The CAM tool needs to be as simple as possible. The graphical display needs to be scalable. For instance, if a prerequisite placeholder (socket) allows for hundreds of concepts from the domain model to be bound to it, it would be impossible to represent these concepts graphically. However, prohibiting such large numbers is not an option, as such a representation might be desirable (for instance, 100 concepts are for beginners, 100 for intermediate and 100 for advanced students). Thus, an alternative is to see part of the graph (with vertical and horizontal scroll bars). Zooming in and out should be allowed. These graphical solutions will be discussed with the visualisation work package. Moreover, if no graphical display is possible, items can be represented textually, e.g. as lists, etc. (For the case above, by double clicking on the 'beginner' labelled socket, the list with the 100 concepts is accessible, for instance).

Requirement arising from this:

R13. The CAM tool should allow for an upper threshold of number of items to be displayed graphically. Scrolls, zooming should be allowed. Alternative text representation should be available for numbers that pass this threshold.

3.3 Integrated authoring experience: the Shell

A key requirement for the authoring environment is that it should make the authoring process easier for the author. As a result of this the CAM model was developed. For the three main layers, the DM, CRT and CAM, there will be a tool implementing these components. If these tools would need to be used separately, the risk exists that the author will be confused and his job will be made more complex. To ensure that the authoring environment indeed makes the authoring process easier, a shell component will be defined. The shell will provide a mechanism for incorporating the DM-, CRT- and CAM-tool and make sure that it appears as one tool for the author, in which he can seamlessly switch between components.

Requirements arising from this:

R05⁴. There should be a common shell for the DM and the CAM tools, and a drag & drop facility from the former to the latter, to allow for one (or more) domain concepts to be selected and dragged into the CAM tool.

⁴ As previously defined in section 1.2, and here, refined.

4 Definition and design of the CAM & CAM tool

4.1 CAM Tool architecture

The CAM tool architecture has a lot of similarities with the DM- and CRT- tool architectures as specified in deliverables 3.1a and 3.2a, as well as according to requirements R02 and R03. The CAM tool is a web-based application schematically shown in the figure below.

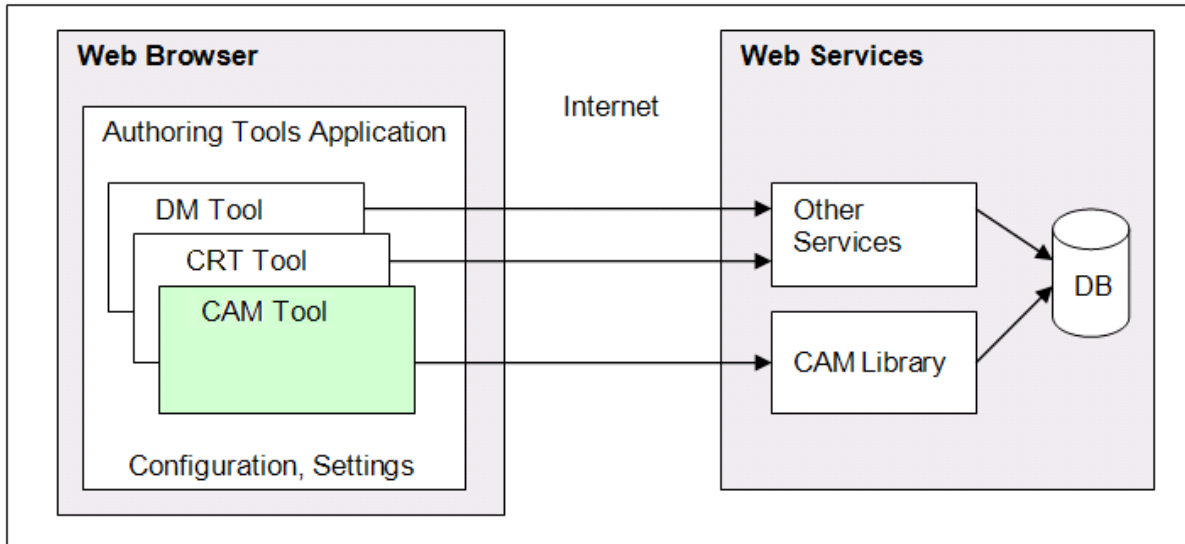


Figure 16. CAM Tool Architecture

The CAM-tool has two components as shown follows:

1. The client part, running in the author's browser, who is authoring a CAM.
This will be a graphical tool, encapsulated in the general GRAPPLE Authoring tool using the shell described in the next section, using the provided common functionality such as configuration and passing information between the different tools (CAM, CRT, DM). Hence each of the tools can be seen as a separate component of the GRAPPLE authoring tool.
2. The server part, running as a Web Service on a web server available for access over the internet. Its main task is storing and retrieving CAM's in a CAM library. It is worth noting that while both the DM and CRT tool employ a similar strategy the Web Services are technically independent from each other. They could even be run in separate locations. However the data is clearly not independent. CAM's will only make sense in combination with the DM's and CRT's they were created with. Hence using the same CAM Web Service in combination with different DM or CRT Web Services can have undesired effects.

4.2 The CAM languages

To describe adaptive behaviour in the scope of the CAM model, the following languages are to be defined:

1. The *CAM visual language*, the drag and drop language for authors to create adaptive story lines,
2. The *CAM internal language*, the XML representation of the CAM visual language, and thus the internal representation of the CAM model instance.
3. The *CAM external language*, for export, based upon the internal representation of the CAM model instance, but with every concept bound.

This apparent profusion of languages will allow the authors with different proficiency levels to describe adaptation, as will be explained in the following sections. The rest of this section will sketch each of these languages in more detail.

The CRT language, (see deliverable d3.2a), is not meant for human consumption, but is a description language of the adaptive behaviour, which will be encapsulated and represented by a graphical symbol in the CAM tool, or by elements such as the *relationship* in the CAM languages. The CRTs are expressed in an XML-based format, containing snippets of GAL (GRAPPLE Adaptation Language) code, see deliverable d1.1a. The GAL strives to be an implementation independent Adaptation Language, and is also intended to be independent of the authoring framework used.

For the scope of the CAM tool and language, the number and type of entities that connect to a CRT instance are important, as well as any restriction on combinations. These will be used to fulfil requirements such as RC1. Moreover, the metadata describing the CRT behaviour in layman's terms is vital for the non-programmer author, to be able use CRTs efficiently in the CAM tool. Thus, the CRT language needs to be able to express, in an easily accessible way, data that is useful for the CAM tool as above.

4.2.1 CAM visual language

The CAM visual language consists of the visual representation of CAM instances, which constitutes the result of mainly the *drag and drop* experience for the author.

The visual representation that effectively is meant for authors to create adaptive story lines expresses the CAM instance in a visual way and is the only language that is intended for the non-programmer author to work with. This language is essential for ensuring the lowest possible threshold in authoring of adaptation.

The CAM languages are all based upon the well known concept of graphs. Hence, the main components are nodes and links. More specifically the nodes are concepts from a DM instance, or placeholders thereof, and the links are the relationships from the CRT instance. There is an additional grouping operator as well.

So to summarise we have the following components that make up the graphical representation:

- *DM Concepts*, as in the DM, or placeholders thereof, with default representation:

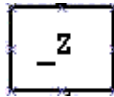


Figure 17. Placeholder for a concept

- *CRT instances*, with a gif based representation and a number of anchor hook locations, where sockets can be hooked into. The default representation is:



Figure 18. Example of prerequisite CRT representation in CAM visual language

- Groups (visually shown as *sockets*) of concepts of placeholders (groups can have anchors to, not shown here). The default representation is:

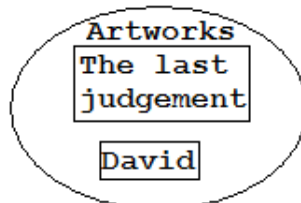


Figure 19. Grouping of concepts

All of these defaults can be replaced by images included from a library, as well as by simple descriptors, as defined by the CAM internal language (see next section and Appendix 3). For more examples of CAM instance descriptions in the visual representation we refer to sections 3.1, 3.2.

4.2.2 CAM internal language

The CAM internal, language consists of the following parts:

A representation of the purely visual aspects; it would be highly confusing if a model when he opened later, looked different from when it was created. Therefore, there is a need to capture the representation of a model in a format. The format is XML-based and is still being extended. More details will be given in deliverable D3.3b.

A format that represents the CAM model instance, hence all (links to) DMs and CRTs used and how they are instantiated. This representation still allows for placeholder concepts, with certain conditions defining which concepts are meant. As we will see in the next section, when the CAM model instance is exported, all such placeholders will be replaced by their concrete matching concepts.

4.2.2.1 XML representation of visual representation.

In order not to confuse the author, in all three tools, the visualisation and layout should be consistent between sessions and the DM, CRT and CAM tools. This means that a mechanism is needed to store and retrieve the graphical representation of the model, presented earlier. The ultimate decision will be made at a later stage and detailed in D3.3b, but something along the lines of SVG [34], that is able to deal with shapes and positions will be needed. The description in this language will be read from DM and CRT tools, in order to represent concepts or placeholders thereof, and CRTs, at the import stage.

Appendix 3 shows the current proposal for the CAM internal language, as well as an example of use. Summarizing, the schema in the appendix allows for defining the following visual information:

- For CRT instance representation: shape, colour and relative position in the CAM instance, or an image representation from the library.
- For socket (placeholder for concepts) representation: shape, colour, size and relative position to the CRT, or an image representation from the library.
- For entity (concept) representation: shape, colour, size and relative position in the socket, or an image representation from the library.
- Moreover, all of the above also allow for a default, which is given by the CRT instance specifications.

4.2.3 CAM external language

The CAM external language is the main format for the output of the CAM tool. It is designed to be used by other systems than the GRAPPLE authoring tool, whilst they interface with the CAM tool. The main use is by the engine-independent compiler shown in Figure 1.

Prior experience [33] shows that non-programmer authors prefer not to be at all involved at the level as described by this language. On the other hand, programmers or authors with a Computer Science background prefer the more 'hands-on' experience. Thus, for the latter authors only, the CAM XML language could be potentially used directly to describe adaptive behaviour. Also prior research showed that an XML-based language is preferable, as it is both more portable, and perceived as easier to manipulate than a pure programming language [12], [33].

All the graphical display information of the internal language is not relevant for the export, and thus removed from the external language. Moreover, the language only needs to assign CRT placeholders of CRTs to domain model concepts, or to labels of domain model concepts, or to instances of resources.

Appendix 3 shows the current proposal for the CAM external language, as well as an example of use.

4.3 Definition and design of the GRAPPLE Authoring Shell

The GRAPPLE Authoring tools, the CAM tool, the CRT tool and the DM tool need an overall umbrella under which to function. Moreover, as a similar look and feel is required (R02, R03) between these tools, and the transition between tools need to be as transparent as possible.

The GRAPPLE Authoring tool is a tool in which authors will be able to, ultimately, specify CAM instances. The author should be able to define all layers of their specific CAM instance with this tool. However, these CAM instances depend on CRT instances and DM concepts.

Thus, we have decided to split the tool into three components corresponding to the mandatory layers (requirement GR1); DM editing; CRT editing and CAM instance editing (adaptive behaviour editing). A shell is defined to integrate these components into a single overall tool.

The user interface will look like the one described in D3.1a and will be described in more detail in D3.3b.

This section describes the common infrastructure for this integration. The UML diagram is depicted in Figure 20. The shell provides three main parts:

- The *Shell* class that implements the actual shell;
- The *AbstractTool* class, an abstract class that tools need to be defined as subclass of, in order to be able to work with the shell;
- The configuration file, *config.xml*, where configuration settings for the tools are stored, such as the content -, CRT - and CAM instance repositories.

For more details see appendix A, which also details a first set of methods of this tool. It was decided that communication between the components of the GRAPPLE authoring tool will be based on services. A number of such services will be defined (e.g., a service for the generation of an unique identifier, etc.) The description of these services will be provided in a different document to the implementers of workpackage WP3 and will guide them before D3.3b is available. It's final form will be included in D3.3b.

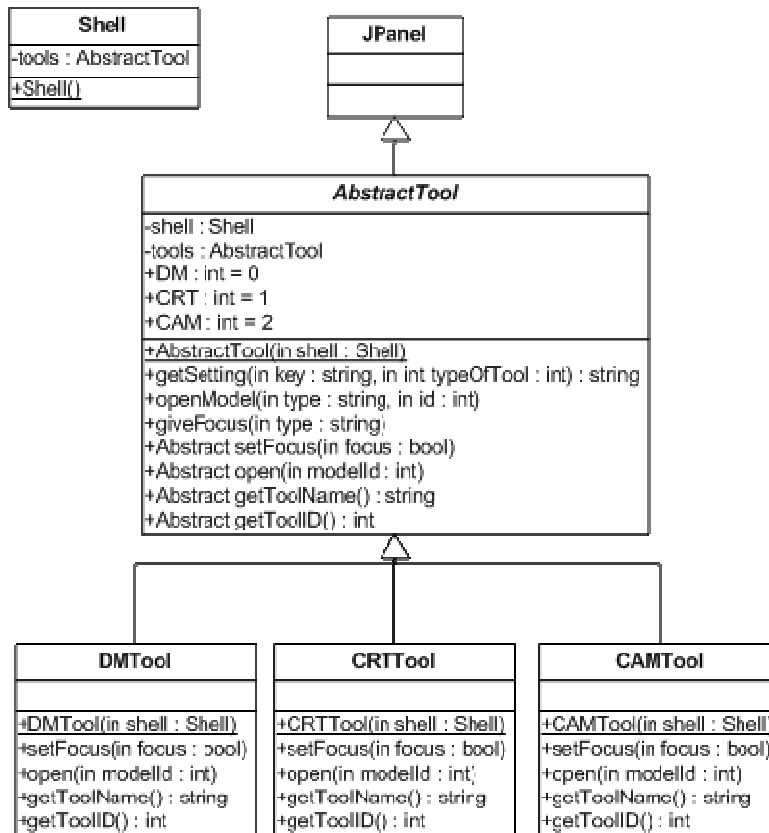


Figure 20. Shell UML diagram

5 Extensions to the CAM tool

5.1 Adaptive Simulations

An adaptive simulation can be represented as 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. The CAM, or more generally, the *Adaptation Model* (AM), is the workspace into which everything is combined to produce an adaptive course or adaptive simulation. Within the GRAPPLE project and this deliverable (D3.3a), it is referred to as the *Conceptual Adaptation Model* (CAM). However, within the context of adaptive simulations, this overly emphasizes the role of concepts. For this reason we drop the 'conceptual' qualifier in these subsections and related deliverables (D3.5a, D3.5b, D3.5c, D4.1a, and D4.1b) to show that it can also be a service-based AM. By a service, we mean something that a user can interact with and not just passively read or look at, e.g. a portlet or applet. The CAM and CAM Tool can be extended to handle services, service relationship types (SRTs), and concept-service relationship types (CSRTs). These are explained in more detail in D3.5a.

CRTs, SRTs, and CSRTs are instantiated in an AM. They are 'dragged' into the workspace by the teacher and bound to specific concepts and services. They can only be instantiated if the concepts and services that the teacher is trying to bind them to match their parameters and the constraints are satisfied. An AM is the teacher'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. The extended schema for adaptive simulations can be found in D3.5a. This XML will specify, amongst other things, the AM's name or identifier, author, and a graph-based model of its contents (concepts, services, CRTs, SRTs, CSRTs, etc.).

5.1.1 Simulation-based extensions to the Adaptation Model

Adaptive simulations suggest new adaptive features (what we can adapt) and adaptive dimensions (what we can adapt to). We can adapt to model fidelity, e.g. time, advice, etc. For example, if a learner is struggling with some concept or service, the simulation can slow the pace of the process and explain it in more detail. Alternatively, if the learner is progressing well, the simulation can speed up the process and provide less artificial instruction. Adaptive workflow completely replaces adaptive navigation. The learner's path through a hyperspace of documents is analogous to their progress through a workflow of processes. We can adaptively vary the level of feedback provided by the system. This feedback can be either natural or artificial and immediate or delayed. We use immediate corrective feedback, even if it is unnatural, when a learner first begins using a simulation or when the simulation's purpose is initial presentation and guidance. In contrast, when a more advanced learner uses the simulation, especially when it is used in practice or as a test, it should provide natural feedback as much as possible, whether delayed or immediate.

In simulations, we replace navigation through hyperlinks with many different types of actions [1], e.g. making a choice, manipulating an object, reacting to an event, and collecting information. Learner actions can be categorized as desirable (requiring natural feedback), neutral, negative (requiring immediate, possibly artificial feedback), or critical. We can also adapt to the learner's role (permission, maintenance, trouble-shooter).

We can enumerate what is required in order to adequately express adaptive simulations and their behaviour. It will need to provide for simulation-specific adaptive features and simulation-specific adaptive dimensions. 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 Resource Model (RM), can fulfil the requirements of a service in the Service Domain Model (SDM). 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. 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.
- 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.
- 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.

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.

5.1.2 Simulation-based extensions to the Adaptation Model Tool

The Adaptation Model Tool, or CAM Tool, provides a graph-based user interface. A *vertex* can represent either a concept or a service. An edge represents a CRT, SRT, or CSRT. Any subgraph can be contracted to a single vertex. It can be expanded again as necessary. This subgraph or contracted vertex represents an activity. An entire graph represents an adaptive simulation. A teacher creates an adaptive simulation by dragging vertices and edges from a palette and adding them to a graph. For example, adding a vertex that represents some concept and linking it with an edge to some service indicates that the concept and the service are related. The service might be an implementation of the concept or it might adhere to the principles of the concept, etc. The type of relationship will be indicated by the type and labelling of the edge. Vertices and edges can also be removed. The actual drawing or layout of a graph does not concern the semantics of the adaptive simulation. It is purely an aid for the teacher when comprehending the structure and contents of an adaptive simulation. The interface enforces constraints during author-time. For example, when dragging a CRT from a palette and adding it to a graph, the edge can only be added between vertices that match the CRT's parameters and does not violate any of its constraints. In particular, a CRT can only be added between concept vertices and an SRT can only be added between service vertices.

The authoring tool for adaptive simulations will share the "GRAPPLE shell". This shell integrates the DM, CRT and CAM tools so that they appear as one tool and the teacher can seamlessly switch between them. For adaptive simulations, we will extend the DM tool to handle the notion of services, the CRT tool to handle SRTs and CSRTs, and the CAM tool to handle the new combinations that services and these new relationship types afford.

5.1.3 A simulation-oriented illustrative scenario

Deliverable D3.5a presents several scenarios to illustrate the construction of an adaptive simulation. We summarize one of the simpler scenarios here, namely the peer review process. The LADiE project [26] has developed a broad set of use case scenarios for activity-based e-Learning. One such scenario requires students to review a set of resources relating to a specific topic. This review is followed by a discussion on the topic, which is guided by the teacher. The students then write and submit a report based on their discussion. The list of steps is:

- 1) Teacher briefs students on the activity.
- 2) Students log into system and access the resources.
- 3) Students discuss the problem.
- 4) Students write report.
- 5) Students submit report.
- 6) System notifies teacher that report has been submitted.

We wish to adapt both the content and the workflow of these activities to the needs of the students. The content can be adapted based on various adaptive axes such as the students' prior knowledge. In addition to this, the services that also make up the activity can be tailored to the needs of the students as well as to their context. For example, the discussion between groups of students can be supported by various different services. A bulletin board service might be appropriate if the activity is expected to run over a long period of time while a chat room service might be more appropriate for a relatively short activity where all of the students are online. Similarly, the authoring and submission tasks can employ a choice of services.

To construct an adaptive simulation for the peer review process, the teacher will drag and drop concepts (e.g. 'problemStatement' and 'report') and services (e.g. 'discussion', 'writeReport' and 'submitReport') into the AM and instantiate pedagogical relationships behind them (e.g. 'prerequisite' and 'andSplit'⁵). This mirrors the construction of an adaptive hypermedia course, as described in the remainder of this deliverable, the only difference being the addition of services and the relationships between them.

⁵ This is an example of an SRT. It indicates that two or more processes can be performed in parallel with the learner occasionally switching between them, e.g. 'discussion' and 'writeReport'. For more details, see deliverable D3.5a.

5.2 Virtual Reality

In the GRAPPLE project, a dedicated authoring tool for VR learning material is foreseen (WP3, Task 3.4). The purpose of the scenarios given here is to explain the relation between the CAM tool and the VR specific authoring tool, and to derive requirements for the CAM tool from the viewpoint of the Virtual Reality authoring tool (described in D3.4a).

5.2.1 A VR-based illustrative scenario

The scenarios given here are based on the ones given in deliverable 3.4a (which describes the design of the VR authoring tool). For a more detailed description of the scenarios we refer to this deliverable.

A teacher wishes to teach the solar system to his students. He feels that describing the solar system only by means of text and illustrations is not appealing enough for the students that he envisages. Therefore, next to plain text explaining the solar system from a scientific point of view, he also wants to provide 3D material.

For the creation of the course, the author has created, using the Domain Model tool of the general authoring tool, the Domain Model (DM) containing concepts such as: Solar System, Sun, Planet, Satellite, Moon, Rotation, Structure, Orbit, Jupiter, Earth, etc.; and relationships between these concepts like Sun is-part-of Solar System, Jupiter is-a Planet and domain specific relations such as Planet rotates-around Sun, and Structure of Planet.

5.2.1.1 Scenario 1: VR-resources for individual concepts

The teacher has some 3D models of planets and satellites and he wants to include them in the course. Therefore, the author will connect these resources to the corresponding concepts in the DM. For instance, he associates an X3D file containing a 3D representation of Jupiter with the DM concept Jupiter; for earth he has several 3D models (X3D format), which he all connects to the DM concept Earth. He makes sure that all these VR-resource are associated with a label "VR". The Domain Model tool of the general authoring tool should supports all this functionality.

Next, the author needs to create the adaptive story line of the course. For most of the story line he can use the general CRTs available in the general CRT Component. E.g., he expresses that all the is-a relationships from his DM should be treated as prerequisite-relations from the CRT tool, i.e. the supertype is a prerequisite for the subtype (like Satellite is a prerequisite for Moon). Furthermore, he also wants to express that a 3D representation of a concept will only be shown when the learner already has some knowledge about the concept. For this, he selects a CRT that allows him to select the type of resource (if the rule is not yet defined in the CRT tool, he has to define it first), e.g.,

- If user knowledge about <X> is above <level>
then present resource for <X> where label = "VR"

Because, the author has different 3D representations for earth, he also wants to be able to specify which resource to use in which learning context. Therefore, he will use a CRT that allows him to select a specific resource, e.g.,

- If <condition> for <X>
then present resource for <X> = <resource>

As follows:

1. If visit(Structure) for Earth
then present resource for Earth = Earth Resource 1
2. If visit(Rotation) for Earth
then present resource for Earth = Earth Resource 2

Where "Earth Resource 1" and "Earth Resource 2" are actual VR-learning resources. Note that it is also possible to select resources based on user knowledge.

In this scenario, there is no need to use the specific VR authoring tool. The VR adaptation needed here is mainly limited to the selection of resources. This results in the following requirements for the CAM tool:

R-VR1: The CAM tool should allow the author to select learning resources based on their type. In particular, it must be possible to specify that a VR learning resource must be selected.

R-VR2: The CAM tool should allow the author to select individual (VR) learning resources. In particular, it must be possible to specify that a particular (VR) learning resource must be selected.

5.2.1.2 Scenario 2: A truly virtual learning world

The teacher has at his disposal a complete virtual 3D environment about the solar system displaying the sun and the different planets, as well as their behaviours, and he would like to create a true adaptive VR course for the solar system starting from this VE. He wants the learner to be able to navigate through the VE, interact with the objects (sun and planets) in the VE, view textual explanations associated with the different objects and watch their behaviours (rotation, orbit, etc.). To avoid that the learners don't know what to do or where to start, as well as to avoid that learners play too much with the objects, the author wants that the environment adapts according to the knowledge of the learner and the material studied. For instance, two possible adaptations could be done: when a learner has finished all study material about Jupiter, the virtual planet will be marked and interaction with the virtual planet will be disabled; when studying the rotation of the planets around the sun, the student will be able to observe this behaviour in the VE; otherwise this behaviour should be disabled.

To achieve this, the author performs the following steps:

1. Similar as in scenario 1, the author will connect all the VR resources needed (i.e. the complete VE as well as all individual resources needed inside the VE and possible alternative resources that he wants to use in the adaptation) to the corresponding concepts in the DM.
2. The author then decides to specify adaptation for the VE at hand and starts the VR specific authoring tool for this resource.
3. This VR authoring tool will allow him to upload the corresponding DM.
4. He will then create the adaptive story line, by using VR-specific CRTs that will be available in the VR specific authoring tool. The format of these CRTs will be similar as the normal CRTs except that their adaptive behaviours are expressed in terms of adaptive behaviour specific for VR. Examples of such rules could be:

```
If concept "X" is a prerequisite of concept "Y",
then mark "X" by spotlight and "Y" by highlight
If first visit
then navigationWithRestrictedInteraction
```

So, in this scenario, the CAM tool will be specified using the specific VR authoring tool. To store and retrieve information from the CRT library, the DM library and the CAM tool repository, the VR authoring tool will use the same web services as the general authoring tool. Therefore another requirement for the CAM tool from the perspective of the VR authoring tool is to provide a number of services that the VR Authoring tool can use to maintain the CAM instances created.

R-VR3: The CAM tool should provide services that can be used by the VR authoring tool to store and retrieve CAM instances.

5.2.1.3 Scenario 3: A hybrid scenario

In this scenario, the author wants to use a combination of classical learning material and a VE. While in scenario 2, the actual learning environment is the VE and all adaptation is performed inside this VE, in this scenario the main learning environment is a classical non-VR environment where adaptation can happen and the VE needs to respond to these adaptations. We illustrate this again with the solar system example. The author wants to offer a combination of classical learning resources and the VE described in scenario 2. He also wants to specify that when the learner starts he will be offered a number of textual learning resources as well as the VE. Inside the VE, the name of the planets already studied should be annotated with a green colour, while the names of the ones still to be studied should be annotated with the red colour.

Next, the author also wants to specify that when the learner is studying a concept (e.g., the sun) (using the textual learning material), then a spotlight should be placed on this object in the VE.

To realize this scenario, the author has to perform the following steps:

1. Similar as in scenario 1 and 2, the author will connect all the VR resources needed (i.e. the complete VE as well as all individual resource needed inside the VE and possible alternative resources that he wants to use in the adaptation) to the corresponding concepts in the DM. This can be done with the general DM tool.
2. For defining the adaptive story line for the classical learning environment, he will use the general CAM tool. There, a rule needs to be used to trigger the presentation of the specific virtual environment resource. For example:

```
If visit(SolarSystem) then present resource for SolarSystem = Solar System VE
```

3. To specify the adaptation inside the VE "Solar System VE", the author needs to use the VR-specific authoring tool in a similar ways as in scenario 2. Using this tool, the author will add the VR specific adaptations.

From this scenario, the following additional requirements can be derived.

R-VR4: The CAM tool should provide services that can be used by the VR authoring tool to add VR specific specifications to an existing CAM instance.

6 Conclusions

In this deliverable we have outlined the definition and design of the Conceptual Adaptation Model (CAM) and CAM tool, as well as the GRAPPLE authoring shell, containing the CAM tool, the CRT tool and the DM tool.

The objective of the *CAM model* is to be a *system-independent high-level model for straightforward, intuitive graphical authoring model* of Adaptive Hypermedia. The objective of the *CAM tool* is to be a tool that illustrates the CAM model, thus allowing for system-independent, high-level modelling for straightforward, intuitive graphical authoring of Adaptive Hypermedia.

The CAM model is formed of three sub-models:

1. the graphical representation (thus graphical language) of the Conceptual Adaptation Model (the *CAM graphical language*). This language expresses how the domain concepts and CRT instances are appearing on the author's screen.
2. the generic, XML-based of the Conceptual Adaptation Model (the *CAM internal language*). This language is the XML-equivalent of the CAM graphical language, and describes how domain concepts and CRT instances are linked together, as well as how they are graphically represented in the author's interface.
3. the generic, XML-based export language of the Conceptual Adaptation Model (the *CAM external language*). This language describes only how domain concepts and CRT instances are linked together, without any graphical information. The CAM external language cannot (normally) be interpreted by an engine by itself, but needs to be paired with the CRT language.

The CAM tool and the GRAPPLE authoring shell are essential components in the GRAPPLE tool kit. This document makes it possible to start the implementation of both, as well as to start the discussion of services used for interaction between the CAM tool and the other two GRAPPLE authoring shell components, the CRT tool and the DM tool, as well as the interaction with the two extensions of the authoring toolkit, for simulation and virtual reality.

7 References

1. Alessi, S., and Trollip, S., *Multimedia for Learning: Methods and Development*. Allyn and Bacon, 3rd edition, 2000.

2. Brown, E., Cristea, A.I., Stewart, C., and Brailsford, T. Patterns in Authoring of Adaptive Educational Hypermedia: A Taxonomy of Learning Styles, International Peer-Reviewed On-line Journal "Education Technology and Society", Special Issue on Authoring of Adaptive Educational Hypermedia, Volume 8, Issue 3. 2005
3. Brusilovsky, P. Developing adaptive educational hypermedia systems: From design models to authoring tools. Authoring Tools for Advanced Technology Learning Environment. Dordrecht. 2003
4. Calvi, L., Cristea, A.I., Towards Generic Adaptive Systems: Analysis of a Case Study, AH 2002, Adaptive Hypermedia and Adaptive Web-Based Systems. 2002
5. Cannataro, M., and Pugliese, A., "XAHM: an XML-based Adaptive Hypermedia Model and its Implementation", 3rd Workshop on Adaptive Hypertext and Hypermedia (AH2001) in conjunction with Twelfth ACM Conference on Hypertext and Hypermedia, Aarhus, Denmark, 2001.
6. Cristea, A.I., Evaluating Adaptive Hypermedia Authoring while Teaching Adaptive Systems, SAC'04, ACM Symposium on Applied Computing, Nicosia, Cyprus. 2004
7. Conlan, O., Wade, V.P., Evaluation of APeLS - An Adaptive eLearning Service based on the Multi-model, Metadata-driven Approach, AH 2004, International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems, Eindhoven, The Netherlands. 2004
8. Cristea, A.I., Adaptive Course Creation for All, ITCC'04, International Conference on Information Technology, Las Vegas, US, IEEE. 2004
9. Cristea, A.I., Stewart, C., Ashman, H. and Cristea, P., Evaluation of Adaptive Hypermedia Systems' Conversion, HT'05, Salzburg, Austria. 2005
10. Cristea, A.I., Calvi, L., The three Layers of Adaptation Granularity, UM'03, International Conference on User Modelling, Pittsburgh, US. 2003
11. Cristea, A.I., Cristea, P., Evaluation of Adaptive Hypermedia Authoring Patterns during a Socrates Programme Class, International Peer-Reviewed On-line & Print Journal "Advanced Technology For Learning" 1(2), ACTA Press. 2004
12. Cristea, A.I., De Bra, P., Explicit Intelligence in Adaptive Hypermedia: Generic Adaptation Languages for Learning Preferences and Styles, HT 2005 CIAH Workshop, Salzburg. (2005)
13. Cristea, A.I., de Mooij, A., LAOS: Layered WWW AHS Authoring Model and their corresponding Algebraic Operators, WWW'03, The Twelfth International World Wide Web Conference, Alternate Track on Education, Budapest, Hungary 2003
14. Cristea, A.I., De Mooij, A., Evaluation of MOT, an AHS Authoring Tool: URD Checklist and a special evaluation class, CATE'03, International Conference on Computers and Advanced Technology in Education, Rhodes, Greece. 2003
15. Cristea, A.I., Verschoor, M., The LAG Grammar for Authoring the Adaptive Web, ITCC'04, International Conference on Information Technology, Las Vegas, US, IEEE. 2004
16. Dagger, D., Wade, V.P., Colan, O., Developing Adaptive Pedagogy with the Adaptive Course Construction Toolkit, (ACCT), AH 2004, Adaptive Hypermedia and Adaptive Web-Based Systems. 2004
17. De Bra, P., Smits, D., Stash, N. (2006). The Design of AHA! , ACM Conference on Hypertext and Hypermedia, pp. 133, Odense, Denmark. 2006
18. De Bra P., Brusilovsky, P., Conejo, R., Multi-Model, Metadata Driven Approach to Adaptive Hypermedia Services for Personalization, AH 2002 Adaptive Hypermedia and Adaptive Web-Based Systems. 2002
19. Freire, M., Rodriguez, P., Comparing Graphs and Trees for Adaptive Hypermedia Authoring, A3EH: Third International Workshop on Authoring of Adaptive and Adaptable Educational Hypermedia, at 12th International Conference on Artificial Intelligence in Education AIED. (2005).
20. Garzotto, F., and Cristea, A.I., ADAPT: Major design dimensions for educational adaptive hypermedia. ED-MEDIA 2004 Conference, 2004
21. Gutierrez, S., Authoring of Adaptive Sequencing for IMS-LD , A3EH, 5th Adaptive Authoring for Educational Hypermedia, Workshop AH 2007, Adaptive Hypermedia and Adaptive Web-Based Systems, Corfu, Greece. 2007

22. Halasz, F., and Schwartz, M. (1994). The Dexter hypertext reference model: Hypermedia. *Communications of the ACM*, 37(2), 30-39.
23. Hendrix, M., Cristea, A.I. and Joy, M., Evaluating the automatic and manual creation process of adaptive lessons, ICALT 2007, IEEE International Conference on Advanced Learning Technologies, Niigata, Japan. 2007
24. Hendrix, M., Cristea A., Nejd, W., Authoring Adaptive Educational Hypermedia on the Semantic Desktop, *International Journal of Learning Technology*, IJLT. 2007
25. Jesus G. Boticario & Olga C. Santos, A dynamic assistance approach to support the development and modelling of adaptive learning scenarios based on educational standards, A3EH, 5th Adaptive Authoring for Educational Hypermedia, Workshop AH 2007, Adaptive Hypermedia and Adaptive Web-Based Systems, Corfu, Greece. 2007
26. JISC. Learning Activity Design in Education: LADiE. Technical report, The Joint Information Systems Committee (JISC), 2006. <http://www.jisc.ac.uk/publications>
27. Koch N., Wirsing M. Software Engineering for Adaptive Hypermedia Applications. 8th International Conference on User Modeling, Sonthofen, Germany, 2001.
28. Martin, B., Mitrovic, A., and Suraweera, P., Domain Modelling with Ontology: A Case Study, A3EH, 5th Adaptive Authoring for Educational Hypermedia, Workshop AH 2007, Adaptive Hypermedia and Adaptive Web-Based Systems, Corfu, Greece. 2007
29. Meccawy, M. Stewart, C. and Ashman, H., Adaptive Educational Hypermedia Content Creation: A Web Service based Architecture, A3EH, 5th Adaptive Authoring for Educational Hypermedia, Workshop AH 2006, Adaptive Hypermedia and Adaptive Web-Based Systems, Dublin, Ireland. 2006
30. Muñoz, F., Ortigosa, A., An Adaptive Course on Template-based Adaptive Hypermedia Design, A3EH, 5th Adaptive Authoring for Educational Hypermedia, Workshop AH 2006, Adaptive Hypermedia and Adaptive Web-Based Systems, Dublin, Ireland. 2006
31. Muntean, C.H., Muntean, G., McManis, J., Cristea, A.I., Quality of Experience-LAOS: create once, use many, use anywhere, *International Journal of Learning Technology*, Special Issue on Authoring of Adaptive and Adaptable Hypermedia, Vol. 3, Issue 3.
32. Ohene-Djan J. A Formal Approach to Personalisable, Adaptive Hyperlink-Based Interaction. PhD thesis, Department of Computing, Goldsmiths College, University of London. 2000.
33. Stash, N., Cristea, A.I., and De Bra, P., Explicit Intelligence in Adaptive Hypermedia: Generic Adaptation Languages for Learning Preferences and Styles, Proceedings of the HT 2005 CIAH Workshop, Salzburg, 2005
34. Scalable Vector Graphics (SVG) 1.1 Specification, W3C Recommendation 14 January 2003, <http://www.w3.org/TR/SVG11/>
35. Vialardi, C., Bravo, J. and Ortigosa, A., Empowering AEH Authors Using Data Mining Techniques, A3EH, 5th Adaptive Authoring for Educational Hypermedia, Workshop AH 2007, Adaptive Hypermedia and Adaptive Web-Based Systems, Corfu, Greece. 2007
36. Wu, H. A Reference Architecture for Adaptive Hypermedia Applications, doctoral thesis, Eindhoven University of Technology, The Netherlands, ISBN 90-386-0572-2.

The Shell

Shell class

The Shell provides the following functionality:

- The shell has a list of tools, containing the DM- CRT- and CAM tools.
- It reads the config.xml file and creates the page layout, DM, CRT, CAM tools buttons. The names of the individual buttons depend on what the implementers specify.

Config file

An example config file is shown below:

```
<?xml version="1.0"?>
<GRAPPLEAuthoringSettings xmlns="http://www.grapple-project.org/ GRAPPLEAuthoringSettings"
xml:lang="en" lang="en">
  <tools>
    <tool id ="0" class="DMTool" />
    <tool id ="1" class="CRTTool" />
    <tool id ="2" class="CAMTool" >
      <key0>value</key0>
    </tool>
  </tools>
  <key1>value</key1>
  <key2>value</key2>
  <key3>value</key3>
</GRAPPLEAuthoringSettings>
```

AbstractTool

Below is a description of the methods of AbstractTool.

Methods and fields already implemented/ set by the AbstractTool (callable by the tools):

private Tools list of AbstractTool

public static final int DM = 0;

public static final int CRT = 1;

public static final int CAM = 2;

getSetting(string key, int typeOfTool): string

- returns the value of a certain setting. key is the name of the setting. typeOfTool is the type of tool the setting is for. The method will try to retrieve the setting for the specific type of tool; if this does not exist it will see if there is a setting for the whole CAM-tool with the name, otherwise it will give back the empty string.

openModel(String type, int id)

- opens the model of type with id ID by calling open(int modelID) of the relevant tool
- does NOT change focus automatically
- there is no GUI involved, this method is purely for tool interoperability

- it's the responsibility of the relevant tool to make sure that changes are saved when required (e.g., loss of focus)
- type == (DM, CRT, CAM) see `getToolName()` below

`giveFocus(int type)`

- set the focus to the relevant tool

methods to be extended tools:

`create (Shell shell, List tools)`

- create pane without having focus
- always call `super.create(shell, tools)`

`setFocus(bool focus)`

- set the focus for this tool?
- The Shell guarantees that upon focus change this method will be called
- The method may still be called even if the focus did not change
- It's the tools responsibility that instances are saved upon change of focus. This is necessary as otherwise the other tools will work with a different version.
- Important for the CAM instance: DM/CRT in use may have changed, check needed.

`open(int modelId)`

- The method will be used to open a certain DM, CRT, CAM instance from either the open dialog or any other application within the shell
- The shell does not guarantee that the model currently open is saved, tools should check this upon loosing focus.
- Calling this method does not imply the model currently opened is different from the one requested

`String getToolName()`

- return the name desired in the shell application for the tool DM, CRT or CAM

`int getToolID()`

- return the tool id: DM, CRT or CAM tool respectively, using the provided public static finals

Comprehensive list of all requirements

General Requirements

GR1. The GRAPPLE authoring tool should allow an extensible number of layers, however a DM, UM and AM layer are mandatory.	10
GR2. The GRAPPLE authoring tool should be visual.	10
GR3. The GRAPPLE tool should contain a UM tool/service (matching the mandatory UM layer of GR1).....	10
GR4. The GRAPPLE authoring tool should contain a DM authoring tool/service (matching the mandatory DM layer of GR1).	10
GR5. The authoring tool should contain an AM authoring tool (matching the mandatory AM layer of GR1).	10
GR5.1: The GRAPPLE authoring tool should contain a CRT type tool.	10
GR5.2: The GRAPPLE authoring tool should contain an adaptation strategy (or story line) creation tool, called CAM (conceptual adaptation model).	10

CAM Requirements

R01. The CAM should allow for an extensible number of layers.....	11
R02. DM concepts and relations representation should keep the same look and feel between different layers (DM, CAM layers).	11
R03. CRT relationships and placeholders should keep the same look and feel between the different layers (CRT and CAM layers).	11
R04. The CAM tool should be able to visually represent the different layers (e.g., showing only one type of relationship or showing multiple relationships, but each with a different representation - in the simplest case, colour).....	11
R05. There should be a common shell for the DM and the CAM tools, and a drag & drop facility from the former to the latter, to allow for one (or more) domain concepts to be selected and dragged into the CAM tool.	12
R06. circular CRT relations should be allowed.	19
R07. Conversion of domain relations into CRTs should be possible in the CAM.	20
R08. Pedagogical labeling should be possible in the CAM.....	20
R09. The CAM tool should allow for simple static analysis for possible termination problems, as far as possible without evaluating the GAL code in CRTs	24
R10. The CAM tool should allow for priorities to be set to the various layers.....	24
R11. The CAM tool should allow for checking of combination problems. (This also implies that such information is available to the CAM tool from the DM and CRT tools respectively).	24
R12. The CAM tool should allow for a preview button	25
R13. The CAM tool should allow for an upper threshold of number of items to be displayed graphically. Scrolls, zooming should be allowed. Alternative text representation should be available for numbers that pass this threshold	25

VR Requirements

R-VR1: The CAM should allow the author selecting learning resources based on their type. More in particular, it must be possible to specify that a VR learning resource must be selected.....	35
R-VR2: The CAM should allow the author selecting individual (VR) learning resources. More in particular, it must be possible to specify that a particular (VR) learning resource must be selected.	35
R-VR3: The CAM tool should provide services that can be used by the VR authoring tool to store and retrieve CAM models.	35

Requirements For The Interface

RI1. The CAM tool should interface with the DM tool..... 15

RI2. The CAM tool should interface with the CRT tool..... 15

Requirements For The Environment

RE1. The CAM tool should allow for graphical input..... 15

CAM Languages schemas and examples

1.1 The CAM internal language

1.1.1. Schema

7.1.1.1 Schema Description

Below we show the schema for the CAM external language. To summarize it consists of the following parts:

- One globally unique *identifier*
- One *author*, a globally unique identifier that refers to the author of this CAM instance.
- A *name*, with a language attribute: intended usage is one name per language.
- A *description*, with a language attribute: intended usage is one description per language.
- One *version*, with information about minor and major versioning; this allows for authors to make small changes to the same CAM instance, without needing another global unique identifier.
- A date & time when the CAM instance was *created*.
- A date & time when the CAM instance was *last updated*.
- At least one *domain model instance*, in the included domain model XML format. A CAM instance could, in principle, be based on multiple domain models, that are then all included here.
- At least one *CRT model instance*, in the included CRT model XML format. A CAM instance would be able to combine different CRT instances. However, a CAM instance built out of one CRT only (e.g., a prerequisite relation) is perfectly possible.
- At least one *processed CAM CRT instance* as below:

7.1.1.2 crt

These CAM Relationships mirror the CRT instances, but add information necessary for internal processing in the CAM tool. This information is of visual and other nature, as explained below. They consist of the following:

- One globally unique identifier *uuid*.
- One *author*, a globally unique identifier that refers to the author of this CAM instance.
- A *name*, with a language attribute: intended usage is one name per language.
- A *description*, with a language attribute: intended usage is one description per language.
- One *version*
- A date & time when the CAM instance was *created*.
- A date & time when the CAM instance was *last updated*.
- Any number of *associatedDMcrt*. The names of the domain model relations that the CAM instance author has associated this CRT instance with, can be used for (semi-) automatic instantiation of the CAM instance based on DM concepts & relationships. For instance, an author can decide that all hierarchical relations in the domain model should be interpreted as prerequisite relations in this CAM instance.
- An optional shape element, indicates the shape that represents this CAM CRT instance, if it differs from the default, which is described in the original CRT model.
- An optional image element, represents the CAM CRT instance, if it differs from the default, which is described in the CRT model.
- An optional colour element, indicates the colour that represents this CAM CRT instance, if it differs from the default, which is described in the CRT model.
- An optional position element, relative positioning in the CAM instance, if it differs from the default.

- At least one `camSocket` as described below:

7.1.1.3 camSocket

Sockets are the visual representation of placeholders in the CRT instance. The current specification of the CRT model provides only two types of sockets, 'target' and 'anchor'. However, the `camSocket` model is more general, as follows. Sockets consist of the following:

- The *type* of the socket (covers 'source', 'target', 'anchor' and any other types desired by the CRT or CAM author).
- The *minCardinality* and *maxCardinality* of the socket, i.e. how many elements the socket should at least contain and how many it should at most contain.
- The *caption* of the socket, i.e. the name that is shown with the socket in the CAM tool
- The *socketId*, a unique identifier
- The relative *position* to the CRT image in the CAM instance (if it differs from the default, as received from the CRT model).
- The *size* of the socket (if it differs from the default, as received from the CRT model).
- The *shape* of the socket (if it differs from the default, as received from the CRT model).
- The *colour* of the socket (if it differs from the default, as received from the CRT model).
- At least one *entity*; the number of entities should be in line with what is expressed in the cardinality, and entities are defined as below:

7.1.1.4 Entity

An entity consists of:

- The *entityId*, the unique identifier of the entity in the CRT instance (the identifier of the placeholder; same as used by the GAL code associated with the CRT instance)
 - zero or one *dmId*, the ID of the concept in the domain model that is assigned to the entityID, with
 - zero or more *labels* for the resource of a concept and
 - an optional *location* for a resource
- A number of *relationshipType* elements, where a requirement for a concept to be involved in a certain relationship in the DM model can be expressed (e.g., the entity should have participated in an IS-A relation)
- The relative *position* in the socket instance, if it differs from the default⁶.
- The *size* of the entity in the socket, if it differs from the default.
- The *shape* of the entity in the CAM instance, if it differs from the default.
- The image of the entity in the CAM instance, if it differs from the default.
- The colour of the entity in the CAM instance, if it differs from the default.

Please note that some of the specifications above imply changes in the CRT model definition (initially defined in D3.2a). These refer to defaults in the representation of concept relations (if they are other than arrows), sockets (if they are other than circles) and entities (if they are other than squares) and to socket (CRT placeholder) types allowed. It is necessary that defaults defined in the CRT are maintained in the CAM tool in order to obtain the same look and feel for the GRAPPLE authoring tools. These changes will be reflected in D3.2b.

⁶ Default here and below refers to the definition in the original CRT instance.

7.1.1.5 CAM Internal Language Schema

Thus the actual CAM Internal Language XML schema is as follows:

```
<?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"
  xmlns:dm="http://www.imslobal.org/xsd/imsvdex_v1p0"
  xmlns:crt="...crt namespace ...">

  <import namespace="http://www.imslobal.org/xsd/imsvdex_v1p0"
    schemaLocation="http://www.imslobal.org/xsd/imsvdex_v1p0.xsd"></import>

  <import namespace="...crt namespace ..." schemaLocation="...crt schema location ..."></import>

  <element name="camInternal" type="tns:camInternalType"></element>

  <element name="camInternal" type="tns:camInternalType"></element>

  <complexType name="camInternalType">
    <sequence>
      <element name="identifier" type="tns:camGuid"></element>
      <element name="author" type="tns:camGuid"></element>
      <element name="name" type="tns:camNameType"></element>
      <element name="description" type="tns:camDescriptionType"></element>
      <element name="version" type="tns:camVersionType"></element>
      <element name="created" type="dateTime"></element>
      <element name="lastUpdated" type="dateTime"></element>
      <element name="domainModel" type="dm:vdexType"
        maxOccurs="unbounded" minOccurs="1">
      </element>
      <element name="crtModel" type="tns:crtModelType"
        maxOccurs="unbounded" minOccurs="1">
      </element>
      <element name="crt" type="tns:camCrtType" maxOccurs="unbounded"
minOccurs="1"></element>
    </sequence>
  </complexType>

  <complexType name="camVersionType">
    <attribute name="major" type="int"></attribute>
    <attribute name="minor" type="int"></attribute>
  </complexType>

  <complexType name="crtModelType">
    <sequence>
      <element name="crt" type="tns:camCrtType"></element>
      <element name="version" type="tns:camVersionType"></element>
      <element name="created" type="dateTime"></element>
      <element name="lastUpdated" type="dateTime"></element>
    </sequence>
  </complexType>

  <complexType name="camCrtType">
    <sequence>
      <element name="uuid" type="tns:camGuid"></element>
      <element name="author" type="tns:camGuid"></element>
      <element name="name" type="tns:camNameType"
        maxOccurs="unbounded" minOccurs="1">
      </element>
      <element name="description" type="tns:camDescriptionType"
        maxOccurs="unbounded" minOccurs="1">
      </element>
      <element name="version" type="tns:camVersionType"></element>
      <element name="created" type="dateTime"></element>
      <element name="lastUpdated" type="dateTime"></element>
      <element name="associatedDMcrt" type="string" maxOccurs="unbounded"
minOccurs="0"></element>
      <element name="shape" type="string" maxOccurs="1"
minOccurs="0">
      </element>
      <element name="image" type="string" maxOccurs="1"
minOccurs="0">
      </element>
    </sequence>
  </complexType>
```

```

        <element name="colour" type="tns:camColourType" maxOccurs="1"
            minOccurs="0">
        </element>
        <element name="position" type="tns:camPositionType"
            maxOccurs="1" minOccurs="0">
        </element>
        <element name="socket" type="tns:camSocketType" maxOccurs="unbounded"
minOccurs="1"></element>
    </sequence>
</complexType>

<complexType name="camNameType">
    <attribute name="language" type="string"></attribute>
    <attribute name="name" type="string"></attribute>
</complexType>

<complexType name="camDescriptionType">
    <simpleContent>
        <extension base="string">
        </extension>
    </simpleContent>
    <attribute name="language" type="string"></attribute>
</complexType>

<complexType name="camPositionType">
    <attribute name="x" type="int"></attribute>
    <attribute name="y" type="int"></attribute>
</complexType>

<complexType name="camSocketType">
    <sequence>
        <element name="caption" type="tns:camCaptionType"></element>
        <element name="socketId" type="tns:camGuid"></element>
        <element name="position" type="tns:camPositionType"></element>
        <element name="size" type="int" maxOccurs="1"
            minOccurs="0">
        </element>
        <element name="shape" type="string" maxOccurs="1"
            minOccurs="0">
        </element>
        <element name="image" type="string" maxOccurs="1"
            minOccurs="0">
        </element>
        <element name="colour" type="tns:camColourType"
            maxOccurs="1" minOccurs="0">
        </element>
        <element name="entity" type="tns:camEntityType"
            maxOccurs="unbounded" minOccurs="1">
        </element>
    </sequence>
    <attribute name="type" type="string"></attribute>
    <attribute name="minCardinality" type="tns:camCardinalityType"></attribute>
    <attribute name="maxCardinality" type="tns:camCardinalityType"></attribute>
</complexType>

<complexType name="camCaptionType">
    <simpleContent>
        <extension base="string">
            <attribute name="language" type="string"></attribute>
        </extension>
    </simpleContent>
</complexType>

<complexType name="camEntityType">
    <sequence>
        <element name="entityId" type="tns:camGuid"></element>
        <choice maxOccurs="1" minOccurs="1">
            <sequence>
                <element name="dmId" type="tns:camGuid" maxOccurs="1"
minOccurs="1"></element>
                <element name="label" type="string"
maxOccurs="unbounded" minOccurs="0"></element>
                <element name="location" type="string" maxOccurs="1"
minOccurs="0"></element>
            </sequence>
        </choice>
    </sequence>
</complexType>

```

```

        <element name="label" type="string"
maxOccurs="unbounded" minOccurs="1"></element>
        <element name="location" type="string" maxOccurs="1"
minOccurs="0"></element>
        </sequence>
        <element name="location" type="string" maxOccurs="1"
minOccurs="1"></element>
    </choice>
    <element name="relationshipType" type="string"></element>
minOccurs="0"></element>
    <element name="size" type="string" maxOccurs="1" minOccurs="0">
    </element>
    <element name="shape" type="string" maxOccurs="1" minOccurs="0">
    </element>
    <element name="image" type="string" maxOccurs="1" minOccurs="0">
    </element>
    <element name="colour" type="tns:camColourType" maxOccurs="1" minOccurs="0">
    </element>
</sequence>
</complexType>

<simpleType name="camGuid">
    <xs:restriction base="xs:string">
        <xs:pattern
            value="[a-z0-9]{8}\-[a-z0-9]{4}\-[a-z0-9]{4}\-[a-z0-9]{4}\-[a-
z0-9]{12}" />
    </xs:restriction>
</simpleType>

<simpleType name="camColourType">
    <xs:restriction base="xs:string">
        <xs:pattern
            value="#"[a-f0-9][a-f0-9][a-f0-9][a-f0-9][a-f0-9][a-f0-9]" />
    </xs:restriction>
</simpleType>

<simpleType name="camCardinalityType">
    <xs:restriction base="xs:string">
        <xs:pattern
            value="[\*\+][0-9]+" />
    </xs:restriction>
</simpleType>
</schema>

```

3.1.2 Example of Using the Internal CAM XML Schema

In the example, colours are used for marking IDs, to enable a quick overview of reuse of such IDs.

```

<?xml version="1.0" encoding="UTF-8"?>
<caminternal xmlns:cam="http://www.grapple-project.org"
xmlns:dm=" http://www.imglobal.org/xsd/imsvdex_vlp0 "
xmlns:crt=" ...crt namespace... "
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.grapple-
project.org grapple_cam.xsd ">

    <!-- general information -->
    <identifier>24990964-1bb8-48f0-b9fc-5ede90193c9b</identifier>
    <author>27214759-3a23-41d5-84f8-bddd419483de</author>
    <name language="English">Michelangelo lesson</name>
    <description language="English">This is a lesson about Michelangelo</description>
    <version major="1" minor="5"/>
    <created>2009-01-01T12:00:00Z</created>
    <lastUpdated>2009-01-15T15:30:00Z</lastUpdated>

    <domainModel>
        <vdx orderSignificant="true" profileType="hierarchicalTokenTerms" language="en"
xsi:schemaLocation="http://www.imglobal.org/xsd/imsvdex_vlp0 imsvdex_vlp0.xsd"
xmlns="http://www.imglobal.org/xsd/imsvdex_vlp0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
            <vocabName><langstring language="en">Michelangelo</langstring></vocabName>
            <vocabIdentifier>http://www.giuntilabs.it/michelangelo.xml</vocabIdentifier>
            <term>

```

```

<termIdentifier>cf5de7f5-12b5-4720-92e5-736cac59985b</termIdentifier>
<caption>
<langstring language="en">Michelangelo</langstring>
</caption>
<description>
<langstring language="en">The renaissance artist Michelangelo</langstring>
</description>
<!-- <? possible concept attributes introduced by the author?> -->
<sequence>
<content>
<caption>
<langstring language="en">Michelangelo picture</langstring>
</caption>
<location href="http://en.wikipedia.org/wiki/Image:Michelangelo.jpg" </location>
<label> Picture </label>
<!-- <? resource attributes/metadata?> -->
</content>
<content>
<caption>
<langstring language="en">Michelangelo description</langstring>
</caption>
<location href="/artsartists/michelangelo.html" </location>
<label> Body text </label>
<!-- <? resource attributes/metadata?> -->
</content>
</sequence>
</term>

<term>
<termIdentifier>1ae895ee-493e-4bfa-85e7-7e19957d2d7c</termIdentifier>
<caption>
<langstring language="en">The last Judgement</langstring>
</caption>
<description>
<langstring language="en">The last Judgement by Michelangelo</langstring>
</description>
<!-- <? possible concept attributes introduced by the author?> -->
<sequence>
<content>
<caption>
<langstring language="en">The last Judgement picture</langstring>
</caption>
<location href="http://en.wikipedia.org/wiki/Image:lastJudgement.jpg"
</location>
<label> Picture </label>
<!-- <? resource attributes/metadata?> -->
</content>
<content>
<caption>
<langstring language="en">The last Judgement description</langstring>
</caption>
<location href="/artsartists/lastJudgement.html" </location>
<label> Body text </label>
<!-- <? resource attributes/metadata?> -->
</content>
</sequence>
</term>

<term>
<termIdentifier>ffdae895-ee49-3e4b-fa85-e77e19957d2d</termIdentifier>
<caption>
<langstring language="en">David</langstring>
</caption>
<description>
<langstring language="en">David by Michelangelo</langstring>
</description>
<!-- <? possible concept attributes introduced by the author?> -->
<sequence>
<content>
<caption>
<langstring language="en">David picture</langstring>
</caption>
<location href="http://en.wikipedia.org/wiki/Image:David.jpg" </location>
<label> Picture </label>
<!-- <? resource attributes/metadata?> -->
</content>
<content>
<caption>
<langstring language="en">David description</langstring>

```

```

</caption>
<location href="/artsartists/David.html" </location>
<label> Body text </label>
<!-- <? resource attributes/metadata?> -->
</content>
</sequence>
</term>

<term>
<termIdentifier>19957d2d-ee49-3e4b-fa85-e77effdae895</termIdentifier>
<caption>
<langstring language="en">Artwork</langstring>
</caption>
<description>
<langstring language="en">Artwork</langstring>
</description>
<!-- <? possible concept attributes introduced by the author?> -->
<sequence>
<content>
<caption>
<langstring language="en">Artwork description</langstring>
</caption>
<location href="/artsartists/Artwork.html" </location>
<label> Body text </label>
<!-- <? resource attributes/metadata?> -->
</content>
</sequence>
</term>

<relationship>
<sourceTerm>dae895ee-493e-4bfa-85e7-
7e19957d2d7c</sourceTerm>
<targetTerm>cf5de7f5-12b5-4720-92e5-
736cac59985b</targetTerm>
<relationshipType>is-created-by</relationshipType>
</relationship>
<relationship>
<sourceTerm>dae895ee-493e-4bfa-85e7-
7e19957d2d7c</sourceTerm>
<targetTerm>ffdae895-ee49-3e4b-fa85-
e77e19957d2d</targetTerm>
<relationshipType>is-created-by</relationshipType>
</relationship>

<relationship>
<sourceTerm>dae895ee-493e-4bfa-85e7-
7e19957d2d7c</sourceTerm>
<targetTerm>19957d2d-ee49-3e4b-fa85-
e77effdae895</targetTerm>
<relationshipType>is_a</relationshipType>
</relationship>
<relationship>
<sourceTerm>dae895ee-493e-4bfa-85e7-
7e19957d2d7c</sourceTerm>
<targetTerm>19957d2d-ee49-3e4b-fa85-
e77effdae895</targetTerm>
<relationshipType>is_a</relationshipType>
</relationship>

</vdex>
</domainModel>

<crtModel>
<CRT UUID="e4f90979-ae32-4c9c-b1f2-4f316228f05f">
<NAME>prerequisite</NAME>
<DESCRIPTION>some description about this CRT</DESCRIPTION>
<SHAPE>line</SHAPE>
<COLOUR>blue</COLOUR>
<ENTITY TYPE="ANCHOR">58aae491-344c-4c17-9400-dfdd063aa0b4</ENTITY>
<ENTITY TYPE="TARGET">FFFFFF7f5-12b5-4720-92e5-736cac5abcde</ENTITY>
<CARDINALITY>2</CARDINALITY>
<GAL>
... pieces of GAL code ...
</GAL>
</CRT>
</crtModel>

<crt>
<uuid>e4f90979-ae32-4c9c-b1f2-4f316228f05f</uuid>

```

```

<author>27214759-3a23-41d5-84f8-bddd419483de</author>
<name language="English">Michelangelo lesson</name>
<description language="English">This is a lesson about Michelangelo</description>
<version major="1" minor="5"/>
<created>2009-01-01T12:00:00Z</created>
<lastUpdated>2009-01-15T15:30:00Z</lastUpdated>
<associatedDMcrt>is-created-by </associatedDMcrt>
// only one concept allowed in this socket:
<camSocket type="source" minCardinality="1" maxCardinality="1">
  // the socket will be shown with a caption saying 'source':
  <caption language="English">source</name>
  <socketId>483de 272-3a23-41d5-84f8-14759bddd419</ socketId >
  <position x="5"y="12"/> // the relative position of the socket to the CRT is given
  <size>10</size> // the size of the socket is given
  <shape>circle</shape> // the shape of the socket is given as circle
  <colour>green</colour> // the colour of the socket is given as green
  <entity>
    // here, the placeholder of the CRT is bound to the domain model concept
    <entityId>58aae491-344c-4c17-9400-dfdd063aa0b4</entityId>
    <dmId>cf5de7f5-12b5-4720-92e5-736cac59985b </dmId>
  </entity>
</camSocket>
<camSocket type="target" minCardinality="1" maxCardinality="1">
  <caption language="English">target</name>
  <socketId>483de 272-3a23-41d5-84f8-14759bddd420</ socketId >
  // as for the source socket, display information for the target socket follows
  <position x="5"y="12"/>
  <size>10</size>
  <shape>circle</shape>
  <colour>green</colour>
  <entity>
    <entityId>FFFFF7f5-12b5-4720-92e5-736cac5abcdc </entityId>
    <dmId>dae895ee-493e-4bfa-85e7-7e19957d2d7c</dmId>
  </entity>
</camSocket>
</crt>
</camInternal>

```

1.2 The CAM external language

3.2.1 Schema

7.1.1.6 Schema Description

Below we show the schema for the CAM external language. To summarize it consists of the following parts:

- One globally unique *identifier*
- One *author*, a globally unique identifier that refers to the author of this particular CAM instance.
- A *name*, with a language attribute: intended usage is one name per language.
- A *description*, with a language attribute: intended usage is one description per language.
- One *version*.
- A date & time when the CAM instance was *created*.
- A date & time when the CAM instance was *last updated*.
- At least one *domain model*, in the included domain model XML format. A CAM instance could, in principle, be based on multiple domain models, that are then all included here.
- At least one *CRT model*, in the included CRT model XML format. A CAM instance would be able to combine different CRT instances. However, a CAM instance built out of one CRT only (e.g., a prerequisite relation) is perfectly possible.
- At least one *relationship* as below:

7.1.1.7 Relationship

Relationships show which CRT instance placeholder is bound to which domain model concepts, or to which resources, directly. Relationships consist of the following:

- A unique identifier *uuid*, referring to the identifier of the CRT instance in the CRT model instance.
- At least one *assign* element; an assign element consists of:
 - An *entityID*, the ID of the entity, in the CRT instance (the identifier of the placeholder; same as used by the GAL code associated with the CRT instance)
 - Either of the following:
 - One *dmID*, the ID of the concept in the domain model that is assigned to the entityID, with a number of labels for the resource of a concept and an optional location for a resource
 - At least one *label* for the resource of a concept and an optional location for a resource
 - a *location* of a resource

Please note that in this way, a CRT placeholder can be bound to a specific resource, if needed, as well as to just one label that describes the type of resource expected.

7.1.1.8 CAM External Language Schema

Thus the actual External CAM XML schema is as follows:

```

<?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"
  xmlns:dm="http://www.imsglobal.org/xsd/imsvdex_v1p0"
  xmlns:crt="...crt namespace ...">

  <import namespace="http://www.imsglobal.org/xsd/imsvdex_v1p0"
    schemaLocation="http://www.imsglobal.org/xsd/imsvdex_v1p0.xsd"></import>

  <import namespace="...crt namespace ..." schemaLocation="...crt schema location ..."></import>

  <element name="camExternal" type="tns:camExternalType"></element>

  <complexType name="camExternalType">
    <sequence>
      <element name="identifier" type="tns:camGuid"></element>
      <element name="author" type="string"></element>
      <element name="name" type="string"></element>
      <element name="description" type="string"></element>
      <element name="version" type="tns:camVersionType"></element>
      <element name="created" type="dateTime"></element>
      <element name="lastUpdated" type="dateTime"></element>
      <element name="domainModel" type="dm:vdexType"
        maxOccurs="unbounded" minOccurs="1">
      </element>
      <element name="crtModel" type="tns:crtModelType"
        maxOccurs="unbounded" minOccurs="1">
      </element>
      <element name="relationship" type="tns:camRelationshipType"
        maxOccurs="unbounded" minOccurs="0"></element>
    </sequence>
  </complexType>

  <complexType name="camVersionType">
    <attribute name="major" type="int"></attribute>
    <attribute name="minor" type="int"></attribute>
  </complexType>

  <complexType name="crtModelType">
    <sequence>
      <element name="crt" type="string"></element>
      <element name="version" type="tns:camVersionType"></element>
      <element name="created" type="dateTime"></element>
      <element name="lastUpdated" type="dateTime"></element>
    </sequence>
  </complexType>

  <complexType name="camNameType">
    <attribute name="language" type="string"></attribute>
    <attribute name="name" type="string"></attribute>
  </complexType>

  <complexType name="camDescriptionType">
    <simpleContent>
      <extension base="string">
      </extension>
    </simpleContent>
    <attribute name="language" type="string"></attribute>
  </complexType>

  <complexType name="camCaptionType">
    <simpleContent>
      <extension base="string">
        <attribute name="language" type="string"></attribute>
      </extension>
    </simpleContent>
  </complexType>

  <complexType name="camEntityType">
    <sequence>
      <element name="entityId" type="tns:camGuid"></element>
      <choice maxOccurs="1" minOccurs="1">

```

```

        <sequence>
            <element name="dmId" type="tns:camGuid" maxOccurs="1"
minOccurs="1"></element>
            <element name="label" type="string"
maxOccurs="unbounded" minOccurs="0"></element>
            <element name="location" type="string" maxOccurs="1"
minOccurs="0"></element>
        </sequence>
        <sequence>
            <element name="label" type="string"
maxOccurs="unbounded" minOccurs="1"></element>
            <element name="location" type="string" maxOccurs="1"
minOccurs="0"></element>
        </sequence>
        <choice>
            <sequence>
                <element name="location" type="string" maxOccurs="1" minOccurs="1"></element>
            </sequence>
        </choice>
    </complexType>

    <simpleType name="camGuid">
        <xs:restriction base="xs:string">
            <xs:pattern
value="[a-z0-9]{8}\-[a-z0-9]{4}\-[a-z0-9]{4}\-[a-z0-9]{4}\-[a-
z0-9]{12}" />
        </xs:restriction>
    </simpleType>

    <complexType name="camRelationshipType">
        <sequence>
            <element name="uuid" type="tns:camGuid"></element>
            <element name="assign" type="tns:camAssignType" maxOccurs="unbounded"
minOccurs="1"></element>
        </sequence>
    </complexType>

    <complexType name="camAssignType">
        <sequence>
            <element name="entityId" type="tns:camGuid"></element>
            <choice maxOccurs="1" minOccurs="1">
                <choice maxOccurs="1" minOccurs="1">
                    <element name="dmId" type="string" maxOccurs="1"
minOccurs="1">
                </element>
                    <element name="label" type="string"
maxOccurs="unbounded" minOccurs="0">
                </element>
                    <element name="location" type="string" maxOccurs="1"
minOccurs="0">
                </element>
                </choice>
            </choice>
            <choice maxOccurs="1" minOccurs="1">
                <element name="label" type="string"
maxOccurs="unbounded" minOccurs="1">
            </element>
                <element name="location" type="string" maxOccurs="1"
minOccurs="0">
            </element>
            </choice>
            <element name="location" type="string"></element>
        </choice>
    </sequence>
</complexType>
</schema>

```

3.2.2 Example of the Application of the External CAM XML Schema

Below we show an example, based on the earlier example about Michelangelo. It includes the single domain model and single CRT that it uses. In the example, colours are used for marking IDs, to enable a quick overview of reuse of such IDs.

```
<?xml version="1.0" encoding="UTF-8"?>
<camExternal xmlns:cam="http://www.grapple-project.org"
xmlns:dm=" http://www.imslobal.org/xsd/imsvdex_v1p0 "
xmlns:crt=" ..crt namespace... "
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.grapple-project.org grapple_cam.xsd ">

  <!-- general information -->
  <identifier>24990964-1bb8-48f0-b9fc-5ede90193c9b</identifier>
  <author>27214759-3a23-41d5-84f8-bddd419483de</author>
  <name language="English">Michelangelo lesson</name>
  <description language="English">This is a lesson about Michelangelo</description>
  <version major="1" minor="5"/>
  <created>2009-01-01T12:00:00Z</created>
  <lastUpdated>2009-01-15T15:30:00Z</lastUpdated>

  <domainModel>
    <vdex orderSignificant="true" profileType="hierarchicalTokenTerms"
    language="en"
    xsi:schemaLocation="http://www.imslobal.org/xsd/imsvdex_v1p0
    imsvdex_v1p0.xsd" xmlns="http://www.imslobal.org/xsd/imsvdex_v1p0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      <vocabName><langstring
      language="en">Michelangelo</langstring></vocabName>
      <vocabIdentifier>http://www.giuntilabs.it/michelangelo.xml</vocabIdentifi
      er>
      <term>
        <termIdentifier>cf5de7f5-12b5-4720-92e5-736cac59985b</termIdentifier>
        <caption>
          <langstring language="en">Michelangelo</langstring>
          </caption>
          <description>
            <langstring language="en">The renaissance artist
            Michelangelo</langstring>
            </description>
            <!-- <? possible concept attributes introduced by the author?> -->
            <sequence>
              <content>
                <caption>
                  <langstring language="en">Michelangelo picture</langstring>
                  </caption>
                  <location> href="http://en.wikipedia.org/wiki/Image:Michelangelo.jpg"
                  </location>
                  <label> Picture </label>
                  <!-- <? resource attributes/metadata?> -->
                  </content>
                <content>
                  <caption>
                    <langstring language="en">Michelangelo description</langstring>
                    </caption>
                    <location> href="/artsartists/michelangelo.html" </location>
                    <label> Body text </label>
                    <!-- <? resource attributes/metadata?> -->
                    </content>
                  </sequence>
                </term>

                <term>
                  <termIdentifier>dae895ee-493e-4bfa-85e7-7e19957d2d7c</termIdentifier>
                  <caption>
                    <langstring language="en">The last Judgement</langstring>
                    </caption>
                    <description>
                      <langstring language="en">The last Judgement by
                      Michelangelo</langstring>
                      </description>
                      <!-- <? possible concept attributes introduced by the author?> -->
                      <sequence>
                        <content>
                          <caption>
```

```

<langstring language="en">The last Judgement picture</langstring>
</caption>
<location href="http://en.wikipedia.org/wiki/Image:lastJudgement.jpg"
</location>
<label> Picture </label>
<!-- <? resource attributes/metadata?> -->
</content>
</caption>
<langstring language="en">The last Judgement description</langstring>
</caption>
<location href="/artsartists/lastJudgement.html" </location>
<label> Body text </label>
<!-- <? resource attributes/metadata?> -->
</content>
</sequence>
</term>

<term>
<termIdentifier>ffdae895-ee49-3e4b-fa85-e77e19957d2d</termIdentifier>
<caption>
<langstring language="en">David</langstring>
</caption>
<description>
<langstring language="en">David by Michelangelo</langstring>
</description>
<!-- <? possible concept attributes introduced by the author?> -->
<sequence>
<content>
<caption>
<langstring language="en">David picture</langstring>
</caption>
<location href="http://en.wikipedia.org/wiki/Image:David.jpg"
</location>
<label> Picture </label>
<!-- <? resource attributes/metadata?> -->
</content>
</caption>
<langstring language="en">David description</langstring>
</caption>
<location href="/artsartists/David.html" </location>
<label> Body text </label>
<!-- <? resource attributes/metadata?> -->
</content>
</sequence>
</term>

<term>
<termIdentifier>19957d2d-ee49-3e4b-fa85-e77effdae895</termIdentifier>
<caption>
<langstring language="en">Artwork</langstring>
</caption>
<description>
<langstring language="en">Artwork</langstring>
</description>
<!-- <? possible concept attributes introduced by the author?> -->
<sequence>
<content>
<caption>
<langstring language="en">Artwork description</langstring>
</caption>
<location href="/artsartists/Artwork.html" </location>
<label> Body text </label>
<!-- <? resource attributes/metadata?> -->
</content>
</sequence>
</term>

<relationship>
<sourceTerm>ffdae895-ee49-3e4b-fa85-e77e19957d2d</sourceTerm>
<targetTerm>cf5de7f5-12b5-4720-92e5-736cac59985b</targetTerm>
<relationshipType>is-created-
by</relationshipType>
</relationship>
</relationship>

```

```

        <sourceTerm>dae895ee-493e-4bfa-85e7-
7e19957d2d7c</sourceTerm>
        <targetTerm>ffdae895-ee49-3e4b-fa85-
e77e19957d2d</targetTerm>
        <relationshipType>is-created-
by</relationshipType>
    </relationship>

    <relationship>
        <sourceTerm>dae895ee-493e-4bfa-85e7-
7e19957d2d7c</sourceTerm>
        <targetTerm>19957d2d-ee49-3e4b-fa85-
e77effdae895</targetTerm>
        <relationshipType>is_a</relationshipType>
    </relationship>
    <relationship>
        <sourceTerm>dae895ee-493e-4bfa-85e7-
7e19957d2d7c</sourceTerm>
        <targetTerm>19957d2d-ee49-3e4b-fa85-
e77effdae895</targetTerm>
        <relationshipType>is_a</relationshipType>
    </relationship>

</vdex>
</domainModel>

<crtModel>
    <CRT UUID="e4f90979-ae32-4c9c-b1f2-4f316228f05f">
    <NAME>prerequisite</NAME>
    <DESCRIPTION>some description about this CRT</DESCRIPTION>
    <SHAPE>line</SHAPE>
    <COLOUR>blue</COLOUR>
    <ENTITY TYPE="ANCHOR">58aae491-344c-4c17-9400-dfdd063aa0b4</ENTITY>
    <ENTITY TYPE="TARGET">FFFFF7f5-12b5-4720-92e5-736cac5abcdc</ENTITY>
    <CARDINALITY>2</CARDINALITY>
    <GAL>
    ... pieces of GAL code ...
    </GAL>
    </CRT>
</crtModel>

<relationship>
    <uuid>e4f90979-ae32-4c9c-b1f2-4f316228f05f</uuid>
    // the CRT placeholders in the two sockets are here connected to actual
    // domain model concepts
    <assign>
        <entityId>58aae491-344c-4c17-9400-dfdd063aa0b4</entityId>
        <dmID>cf5de7f5-12b5-4720-92e5-736cac59985b</dmID>
    </assign>
    <assign>
        <entityId>FFFFF7f5-12b5-4720-92e5-736cac5abcdc </entityId>
        <dmID>dae895ee-493e-4bfa-85e7-7e19957d2d7c</dmID>
    </assign>
</relationship>
</camExternal>

```