

GRAPPLE

D3.3b Version: 1.0

Initial Implementation of the Concept Adaptation Model Tool

Document Type	Deliverable
Editor(s):	Maurice Hendrix (Warwick)
Author(s):	Maurice Hendrix (Warwick), Martin Harrigan (TCD)
Reviewer(s):	Noaa Barak (ATOS), Avi Naim (UCAM)
Work Package:	WP3
Due Date:	M21
Version:	1.0
Version Date:	2009-11-11
Total number of pages:	42

Abstract: This deliverable outlines the initial implementation of the Conceptual Adaptation Model (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 CAM will be system-independent and will be a model for straightforward, graphical authoring of Adaptive Hypermedia. The CAM tool has already been described in deliverable D3.3a. In this document, we give an update on the design of the technical specifications for the CAM format and tool as well as for the Shell, integrating the tools into the Grapple Authoring tool (GAT). Finally a user guide is included, to help authors create and modify CAMs.

Keyword list: authoring tool, concept adaptation model, CAM tool, domain model, conceptual adaptation model, concept relationship type

Summary

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 was previously described in deliverable 3.3a and is a tool for creating CAM instances with the help of the DM tool (as described in deliverable 3.1b) and the CRT tool (described in deliverable 3.2b).

This deliverable outlines the initial implementation of the Conceptual Adaptation Model (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. The CAM tool has already been described in deliverable D3.3a. In this document, we give an update on the design of the technical specifications for the CAM format and tool as well as for the Shell, integrating the tools into the Grapple Authoring tool (GAT). Finally a user guide is included, to help content authors create and modify CAMs.

The first prototype is made available online at <http://dyn070.win.tue.nl:8080/GAT/>

Authors

Person	Email	Partner code
Maurice Hendrix	maurice@dcs.warwick.ac.uk	Warwick
Martin Harrigan	martin.harrigan@cs.tcd.ie	TCD

Table of Contents

SUMMARY	2
AUTHORS	2
TABLE OF CONTENTS	2
TABLES AND FIGURES.....	4
LIST OF ACRONYMS AND ABBREVIATIONS	5
1 INTRODUCTION	6
2 SUMMARY OF THE CAM MODEL	6
2.1 AHAM	6
2.2 LAOS	7
2.3 CAM.....	8
2.4 Relations between the layers of the GRAPPLE authoring tool.....	9
2.5 Relations between the CAM and the adaptation engine(s)	9

3	SPECIFICATION OF THE CAM FORMAT	9
3.1	Structure of the CAM format.....	9
3.2	Common header.....	9
3.3	The CAM formats	10
3.4	Updates on the CAM visual language compared to d3.3a	11
3.5	CAM internal language.....	11
3.6	XML Schema of cam internal language.....	13
3.7	CAM external language	15
4	IMPLEMENTATION AND INTEGRATION OF THE CAM TOOL	16
4.1	The GAT shell and Web Service.....	16
4.2	Implementation of the CAM Tool.....	16
4.3	Integration of the CAM Tool with the GAT shell and Web Service	17
5	PRELIMINARY EVALUATION	18
5.1.1	Qualitative evaluation	18
5.1.2	Quantitative evaluation	21
5.2	Conclusions	23
6	USER GUIDE ON THE CAM TOOL	23
6.1	Inserting CRTs	24
6.2	Filling sockets with concepts.....	26
6.3	Editing CRTs and Sockets	29
6.4	Deleting concepts and CRTs	29
7	CONCLUSION AND OUTLOOK	30
	REFERENCES	30
8	APPENDIX	32
8.1	CAM Examples	32
8.2	Information for integrating GAT tools into the GAT shell	38
8.2.1	assets	38
8.2.2	net.....	39
8.2.3	org.grapple_authoring_tool.authoring.events.....	39
8.2.4	org.grapple_authoring_tool.authoring.ui.....	40
8.2.5	org.grapple_authoring_tool.authoring.utils	41

8.2.6	org.grapple_authoring_tool.authoring.CopyPaste.....	42
8.2.7	org.grapple_authoring_tool.authoring.GrappleVgraph.....	42
8.2.8	org.grapple_authoring_tool.authoring.crtshapes.....	42
8.2.9	org.grapple_authoring_tool.authoring.socketshapes.....	42
8.2.10	org.un.....	42

Tables and Figures

List of Figures

Figure 1.	a socket with one concept.....	10
Figure 2.	Example of prerequisite CRT representation in CAM visual language.....	10
Figure 3.	Grouping of concepts.....	10
Figure 4.	GAT Tool Architecture.....	17
Figure 5:	The CAM Tool integrated in the GAT shell.....	18
Figure 6	SUS score GAT.....	22
Figure 7.	Opening the GAT tool.....	23
Figure 8.	Creating a new CAM.....	24
Figure 9.	Inserting a CRT instance into a CAM.....	24
Figure 10.	Choosing a CRT to insert.....	24
Figure 11.	The prerequisite CRT, which has 2 sockets, inserted into the CAM.....	25
Figure 12.	Copy a CRT instance. Other actions are delete and cut (copy then delete).....	25
Figure 13.	Past CRT is now enabled.....	26
Figure 14.	Copy concepts from a socket in a CAM window.....	26
Figure 15.	Pasting Concepts into sockets.....	27
Figure 16.	Socket editing screen providing an alternate place to paste concepts and an interface to insert concepts.....	27
Figure 17.	Selection interface for the Domain Model the concept comes from.....	27
Figure 18.	Selection of the Concept.....	28
Figure 19.	Insert External Location.....	28
Figure 20.	Two sockets containing the concepts Star.....	29
Figure 21.	CRT editing screen.....	29
Figure 22.	Deleting all concepts from a socket.....	30

List of Acronyms and Abbreviations

CAM	Conceptual Adaptation Model; also called: Adaptive Story Line; Adaptation Strategy
CRT	Concept Relationship Type(s); also called: (Pedagogical) Relationship Type(s)
DM	Domain Model
GAL	GRAPPLE Adaptation Language
GAT	GRAPPLE Authoring Tools
GALE	GRAPPLE Adaptive Learning Environment
GUMF	GRAPPLE User Model Framework
GRAPPLE	Generic Responsive Adaptive Personalized Learning Environment
GUI	Graphical User Interface
LAG	Layers of Adaptation Granularity
UM	User Model
UUID	Universally Unique Identifier

1 Introduction

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 illustrate the CAM model, with its specified attributes. The CAM tool is a tool for creating CAM instances with the help of the DM tool (as described in deliverable 3.1a and 3.1b) and the CRT tool (described in deliverable D3.2a and D3.2b).

The main deliverable for the task T3.3b is the integration of the CAM tool into the GAT authoring tool. This document is meant to provide an update upon the design of the CAM tool and the Shell that integrates the DM, CRT and CAM tool into the Grapple Authoring Tool (GAT).

The CAM model has is a system-independent high-level model. The CAM model will support straightforward, intuitive graphical authoring of Adaptive Hypermedia. The CAM tool has already been described in deliverable D3.3a. In this document, we give an update on the design of the technical specifications for the CAM format and tool as well as for the Shell, integrating the tools into the Grapple Authoring tool (GAT). Finally a user guide is included, to help content author create and modify CAMs.

Work package 3 is mainly connected to WP5, WP9, WP7 and WP8.

WP5 explores the suitability of existing standards for expressing the various models developed in GRAPPLE. The interaction between WP5 and WP3 results in an understanding of to what extend the CAM model can be expressed in various inputs, roles, interactions, adaptation methods & techniques within the limits of the explored standards. Specifically D5.3a, b and c performs the matching to the available LMS and IMS-LD.

WP7 is concerned with the integration of all different components into a coherent learning environment. The GAT has to communicate with the GALE (Grapple Adaptive Learning Environment), delivery environment for deploying courses as well as the GUMF (Grapple User Modelling Framework). Hence there is a direct link between this deliverable and D7.1 (a and b) about *the operational infrastructure specification and design* and D7.2 (a, b and c) about *Data models and interoperability*.

WP9 is mainly concerned with gathering requirements, providing documentation for and evaluation the experience in Higher education. WP3, and this deliverable in particular is related to D9.1, *Requirements Specification by Academic Users on Integrated Adaptive Learning Services, First Documentation and Training for GRAPPLE Users*. This deliverable is also related to the evaluations performed in WP8, specifically D8.2 *Evaluation Guidelines*.

The remainder of this deliverable is structured as follows. Chapter 3 describes the CAM format as it has been implemented in the initial prototype of the CAM tool. The implementation of the CAM tool and the integration of all tools into the Grapple Authoring Tool (GAT) (using the shell), is described in chapter 4. Chapter 6 gives a short user guide of the whole GAT, including the CAM tool for the benefit of authors. Finally chapter 7 presents our conclusions and outlook on future work.

2 Summary of the CAM model

This section revisits the design of the CAM model as it has been described in deliverable D3.3a for the benefit of the reader. The CAM model is based upon lessons learnt from the AHAM [13] and LAOS [5] models, as well as being inspired by the multi-model, metadata-driven approach to content adaptation [6], and has incorporated ideas from other models, such as Dexter [8], XAHM [2], Munich [10], UWE [9], ADAPT [7]. 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.

2.1 AHAM

The AHAM [13] reference model for Adaptive Hypermedia Systems (AHS) describes adaptive applications as consisting of three main layers:

- **The Domain Model (DM)** describes concepts, then 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 simple subject domain ontology, except for the concept relationships that have a (pedagogical) meaning.

- **The User Model (UM)** 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)** 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:
 - o 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;
 - o 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 [6], a graphical authoring tool, the Graph Author, is used to define the DM and draw a graph of concept relationships (of different types). As the name “graph” suggests, concept relationships are (unary or) binary, whilst in AHAM there is no restriction to the number of concepts that together may form a relationship.

2.2 LAOS

The LAOS model [5] is a five-layered extension of AHAM, that responds to and improves upon some of the limitations of AHAM, with special focus on authoring and reuse issues. In LAOS, different aspects of the adaptation model are distributed over multiple layers in the model [10], in particular the:

- **domain model (DM)**, similar to the AHAM model DM, with the exception of allowing only domain-specific information (concepts and links), unlike in AHAM;
- **goal and constraints model (GM)**, extracting and concentrating all goal-related (e.g., pedagogical, for educational applications) information previously intermingled with domain information in the AHAM model;
- **presentation model (PM)**, extracting and concentrating all presentation information previously intermingled with domain information in AHAM.
- **user model (UM)** similar to the one in AHAM.
- **adaptation model (AM)**, extending the ideas of genericity and specificity:
 - o generic adaptation rules: instead of connected to a type of event (such as visit), allowing 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.
 - o specific adaptation rules: similar to the generic rules, but applied to a specific concept, identified from a given domain map (this is mainly used for backward compatibility with previous types of adaptive hypermedia, like AHAM, where concepts were connected only in this explicit, specific way, and only reusable with one DM).

In this way, pedagogical information, e.g., can be expressed in the GM alone, and kept apart from other information. Also, the PM describes the final look & feel of the presentation as well as the quality of service parameters (e.g. for mobile devices).

CRT's in LAOS are also of different types, 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 [3] built on LAOS); similarly,
- CRT's in the GM describe only goal-related relations (pedagogic relations for the educational domain, such as AND-OR relations with pedagogical labels, as available in the MOT).

Moreover, the adaptation model allows for different levels (and thus degrees) of reuse of adaptation, by conforming to the LAG framework [4], [11]: establishing as a first level the (event-)-condition-action rules, as in AHAM, that are the building stones for adaptation, and also thus forming the assembly-language type of adaptation specifications; at the second level, allowing for more sophisticated adaptation languages [3] (such as LAG [4], [11] or LAG-XLS [4], [11]) ; finally, at the highest level, adaptation strategies, comprising

reusable, annotated storylines of adaptation and personalization, that can be applied to various domain models.

This type of layered structure allows reuse of each layer separately, beside the type of reuse described above, and thus is more flexible and more appropriate for authoring, where the 'write-once use-many' paradigm is most relevant .

2.3 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.

GR2: The GRAPPLE authoring tool should be visual.

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 (conceptual adaptation model).

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 ensures a high-level, non-programming access to authoring behaviour specification. In this paper, due to its important role in integrating all the other authoring elements, as well as due to the lack of space, the focus is on the CAM model and tool.

Because of the limited nature and specificity of the DM and CRT authoring, and the general requirement GR1, this leads to the following requirements (R) for the CAM authoring tool:

R1. The CAM should allow for an extensible number of layers.

R2. DM concepts and relations representation should keep the same look and feel between different layers (DM, CAM layers).

R3. 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 R2, R3, the requirement R4 can be defined as follows:

R4. 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 structures to actual adaptation rules is described and exemplified in Section 3.

2.4 Relations between the layers of the GRAPPLE authoring 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. This implies the following requirement:

R5. 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. Note: the Google Web Toolkit (GWT) was chosen for this common shell.

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 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.

2.5 Relations between the CAM and the adaptation engine(s)

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 or has converters for one or more of the exported adaptation languages. The CAM tool thus will export several languages of various levels: the CAM XML language, a combination of CAM and CRT language, the adaptation engine's own Grapple Adaptation Language (GAL) language, possibly also other languages, such as the LAG language [4], [11].

3 Specification of the CAM Format

In order to exchange CAMs with other GRAPPLE components a data format has to be defined which can be written by the CAM Tool and read by other components. Most importantly CAMs have to be read by converters to adaptive display engines such as GALE.

CAMs are expressed in XML format and follow an XML schema specification which defines the structure of the XML format.

In this section the common explanation of the CAM format and structure is given, followed by the XML schema definition. Finally it is outlined which changes have been made to the CAM definition since the last Deliverable 3.3a. A complete example of a CAM definition can be found in Appendix 8.1.

3.1 Structure of the CAM format

For the reason of compatibility with the other formats of the GRAPPLE authoring tools, a common wrapper format has been defined. Another benefit is, that the web service can treat the different models transparently the same, without any knowledge on the differences between the types of model (DM, CRT, CAM), and can use the header info for quick look-up of models.

3.2 Common header

The following XML code outlines the common structure of GRAPPLE authoring tool models. Each model has a common header part, where Meta data of the model is located. Meta-data include title, common description, creation- and update time, author, authorisation, and the UUID of the model. Below is a short outline of the common parts of the XML format.

```
<model>
  <header> //the header is shared between all models
    <modeluuid>660e8400-e29b-41d4-a716-446655440000</modeluuid>
    //the modelUuid is a unique identifier for the model.
    <modeltype>DM</modeltype>
```

```
//Types can be DM, CRT, CAM or their VR and Simulation equivalent
<authoruuiid>660e8400-e29b- a716-41d4-446655440000</authoruuiid>
// the author UUID is the unique identifier of the author, who created the model
<authorisation>readwrite</authorisation>
// authorisation for all other author; the original author always has all permissions.
//permissions can be none (no access by other authors) read (read-only access) and
//readwrite (rad and write access)
<creationtime>1242904243000</creationtime> //the timestamp of creation of the model
<updatetime>1242904243000</updatetime> //the timestamp the model was last updated
<title>sun-example-dm</title> //the title of the model
<description>sun-example-dm</description> //the description of the model
</header>
<body>//the body contains the actual content of the model
<cam> //depending on the model there will be a CAM, CRT or dm tag to indicate the model type
//content of the model (in this case cam) see below for more details
</cam>
</body>
</model>
```

3.3 The CAM formats

This section we describe the CAM languages as they were implemented in the first prototype of the GAT. We will also indicate what has been updated compared to the preliminary version in deliverable 3.3a.

3.3.1.1 CAM visual language

The visual representation which 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 sockets, placeholders that can contain sets of concepts from a DM instance, and the links are the relationships from the CRT instance. There is an additional grouping operator as well.

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

- A socket, a placeholder for sets of DM Concepts, with default representation:



Figure 1. A socket with one concept

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



Figure 2. Example of prerequisite CRT representation in CAM visual language

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

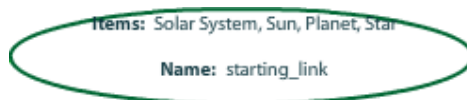


Figure 3. Grouping of concepts

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.

3.4 Updates on the CAM visual language compared to d3.3a

There are a number of updates on the CAM visual language. The main update is that the notion of sockets has been introduced. This notion is similar to what we envisioned as *'grouping of concepts'* in d3.3a. Also the notion of placeholder concepts has been abandoned to an extent. The format does not allow for concepts that are not attached to sockets. As such, there is no representation for placeholder concepts. The sockets act as the placeholders that can contain concepts. The final update is the cosmetic update of the representation of CRTs, concepts and sockets. Please note that although in this version the concepts are merely represented as text in the sockets, a more visual representation, closer to that described in d3.3a, is envisioned for the second prototype, due in Month 30 (July 2010).

3.5 CAM internal language

The CAM internal language is the main format for storage and manipulation of the CAM tool. It is designed to be used by other systems, whilst they interface with the CAM tool. Prior experience with the LAG language [4] 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, as in the development of the LAG-XLS language [4]. As the LAG-XLS language was only aimed at learning styles, however, and we wish to adopt a wider scope of adaptivity, a new language has to be created.

The CAM format can be used to describe, not only direct CRT relations, but also more global information & flow of a course:

- Start and end states can be described with a special CRT. The CRT is different only in that the GAL [12] code simply states "start" or "end".
- Main concepts of a course, useful for visualising the course, can be indicated in the name/meta data of the sockets.
- Learning goals and objectives can be derived from the selection of start and end states and main concepts.

Below we illustrate the new CAM language via an example. We see this type of CAM output as XML descriptions of groups of concepts and *named typed* relations between them. For example, if concept A is to be seen before concept B, a CAM would be:

```
<model>
  <header> .. </header>
<body>
  <cam>
    <camInternal>
      <domainModel>..</domainModel>
      <crtModel> .. </crtModel>
      <crt>
        <uuid>cf5de7f5-12b5-4720-92e5-736cac59985b</uuid>
        <shape>diamond</shape>
        <colour>#C0C0C0</colour>
        <camSocket>
          <uuid>e9b45bd0-6013-11de-8a39-0800200c9a66</uuid>
          <socketId>cf5de7f5-12b5-4720-92e5-zzzzzzzzz</socketId>
          <position><x>100</x><y>250</y></position>
          <size>10</size>
          <shape>rectangle</shape>
          <colour>#006633</colour>
          <entity><dmId>201-de-8a39-0800200c9a66</dmId></entity>
        </camSocket>
      </crt>
    </camInternal>
  </cam>
</body>
</model>
```

```

    <socketId>2b5-4720-92e5-pppppppppppp</socketId>
    <position><x>100</x><y>400</y></position>
    <size>10</size>
    <shape>rectangle</shape>
    <colour>#006633</colour>
    <entity><dmId>11de-8a39-0800200c9a66</dmId></entity>
  </camSocket>
</crt>
</camInternal>
</cam>
</body>
</model>

```

As we can see the CAM language consists of two parts the *header* and the *body*, the header is the common header described above. The *body* part contains the information specific for the CAM model. The body part contains the following information:

- The domainModels and crtModels in use, as described in sections 6.1 and 6.2
- A number of *crt* tags, containing the instantiation of CRTs from *the crtModel* with actual concepts from *the domainModel*.

The *crt* tag contains the following information:

- A unique identifier the uuid
- The shape and colour of the CRT, for displaying the CRT in the CAM tool, this is optional, defaults will be used if the tags are omitted.
- An optional position element. The position of a CRT needs to be given if the CRT is not binary (1 or >2 sockets). For binary CRTs the position is calculated, based on the position of the sockets.
- A number (≥ 1) of camSockets.

The *camSockets* contain the instantiation with the actual concepts, they contain the following information:

- An optional caption, a name for the socket, for the authors' assistance.
- A unique identifier the uuid
- A socketId, the unique id of the specific socket in the crtModel
- A position, the position where the socket should be displayed in the CAM Tool.
- An optional shape and colour, the shape and colour a CAM socket should have, a default will be used if the tags are omitted.
- A number of entity tags.

The entity contains the instantiation with the actual concepts, they contain the following information:

- 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

The following items have not been implemented in the CAM tool yet. At the moment the entities are represented with text. The intention is to have the graphical elements described below available by the next prototype (due month 30, July 2010), if time permits (this is of a lower priority than delivering a working tool). The relationshipType element is an important part of the resource retrieval strategy and will be present in the next prototype.

- 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 default.
- 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

The description of the subject domain and behaviour semantics of the CRT called 'prerequisite' needs to be separately imported from the DM and CRT repositories. These descriptions are then included in the *domainModel* and *crtModel* parts.

3.6 XML Schema of cam internal language

This section contains the XML schema specification of the CAM XML format. CAMs which are created by the CAM Tool are valid against this schema specification. This specification is supposed to be used by all GRAPPLE tools and components which read CAMs. An example of a CAM which implements this specification can be found in Appendix 8.1.

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema" targetNamespace="http://www.example.org/CAM-external-oct2009" xmlns:tns="http://www.example.org/CAM-external-oct2009"
elementFormDefault="qualified" xmlns:Q1="http://www.imsglobal.org/xsd/imsvdex_v1p0"
xmlns:Q2="http://grapple-project.org/GAT/" xmlns:Q3="http://www.grapple-project.org">

  <import schemaLocation="dm-vdex-extensions.xsd" namespace="http://www.grapple-
project.org"></import>
  <import schemaLocation="CRTOct2009.xsd" namespace="http://grapple-project.org/GAT/"></import>
  <import schemaLocation="vdex.xsd"
namespace="http://www.imsglobal.org/xsd/imsvdex_v1p0"></import>
  <element name="model" type="tns:modelType"></element>

  <complexType name="modelType">
    <sequence maxOccurs="1" minOccurs="1">
      <element name="header" type="tns:headerType" maxOccurs="1"
minOccurs="1"></element>
      <element name="body" type="tns:camBodyType" maxOccurs="1" minOccurs="1"></element>
    </sequence>
  </complexType>

  <complexType name="headerType">
    <sequence>
      <element name="modeluuid" type="tns:UUIDtype" maxOccurs="1"
minOccurs="1">
      </element>
      <element name="modeltype" type="string" maxOccurs="1"
minOccurs="1">
      </element>
      <element name="authoruuid" type="tns:UUIDtype" maxOccurs="1"
minOccurs="1"></element>
      <element name="authorisation" type="tns:authorisationType" maxOccurs="1"
minOccurs="1"></element>
      <element name="creationtime" type="dateTime" maxOccurs="1"
minOccurs="1"></element>
      <element name="updatetime" type="dateTime" maxOccurs="1" minOccurs="1"></element>
      <element name="title" type="string" maxOccurs="1" minOccurs="1"></element>
      <element name="description" type="string" maxOccurs="1" minOccurs="1"></element>
    </sequence>
  </complexType>

  <complexType name="camBodyType">
    <sequence>
      <element name="cam" type="tns:camType" maxOccurs="1" minOccurs="1"></element>
    </sequence>
  </complexType>

  <simpleType name="authorisationType">
    <restriction base="string">
      <enumeration value="read"></enumeration>
      <enumeration value="write"></enumeration>
      <enumeration value="readwrite"></enumeration>
    </restriction>
  </simpleType>

  <simpleType name="UUIDtype">
    <restriction base="string">
      <pattern
```

```

        value="[a-z0-9]{8}\-[a-z0-9]{4}\-[a-z0-9]{4}\-[a-z0-9]{4}\-[a-z0-9]{12}">
    </pattern>
</restriction>
</simpleType>

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

<complexType name="camInternalType">
    <sequence>
        <element name="domainModel" type="tns:dmModelType" maxOccurs="1"
            minOccurs="1">
        </element>
        <element name="crtModel" type="tns:crtModelType" maxOccurs="1"
            minOccurs="1">
        </element>
        <element name="crt" type="string" maxOccurs="unbounded" minOccurs="0"/></element>
    </sequence>
</complexType>

<complexType name="positionType">
    <sequence>
        <element name="x" type="int"/></element>
        <element name="y" type="int"/></element>
    </sequence>
</complexType>

<complexType name="camSocketType">
    <sequence>
        <element name="caption" type="string" maxOccurs="1"
            minOccurs="0">
        </element>
        <element name="uuid" type="tns:UUIDtype" maxOccurs="1"
            minOccurs="1">
        </element>
        <element name="socketId" type="tns:UUIDtype" maxOccurs="1"
            minOccurs="1">
        </element>
        <element name="position" type="tns:positionType"
            maxOccurs="1" minOccurs="1">
        </element>
        <element name="shape" type="string" maxOccurs="1" minOccurs="0"/></element>
        <element name="colour" type="string" maxOccurs="1" minOccurs="0"/></element>
        <element name="entity" type="tns:entityType" maxOccurs="unbounded"
minOccurs="0"/></element>
    </sequence>
</complexType>

<complexType name="entityType">
    <sequence>
        <element name="dmID" type="tns:UUIDtype" maxOccurs="1"
            minOccurs="0">
        </element>
        <element name="label" type="string" maxOccurs="unbounded"
            minOccurs="0">
        </element>
        <element name="relationshipType" type="string" maxOccurs="1"
minOccurs="0"/></element>
        <element name="position" type="tns:positionType" maxOccurs="1"
minOccurs="0"/></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="string" maxOccurs="1" minOccurs="0"/></element>
    </sequence>
</complexType>

<complexType name="dmModelType">
    <sequence>
        <element name="model" type="tns:dmModelType2" maxOccurs="unbounded"
minOccurs="0"/></element>
    </sequence>
</complexType>

```

```

<complexType name="dmModelType2">
  <sequence>
    <element name="header" type="tns:headerType"></element>
    <element name="body" type="tns:dmType"></element>
  </sequence>
</complexType>

<complexType name="dmType">
  <sequence>
    <element name="vdex" type="Q1:vdexType" maxOccurs="1" minOccurs="1"></element>
  </sequence>
</complexType>

<complexType name="crtModelType">
  <sequence>
    <element name="model" type="tns:crtModelType2" maxOccurs="unbounded"
minOccurs="0"></element>
  </sequence>
</complexType>

<complexType name="crtModelType2">
  <sequence>
    <element name="header" type="tns:headerType" maxOccurs="1"
minOccurs="1"></element>
    <element name="body" type="tns:crtType" maxOccurs="1"
minOccurs="1"><complexType></complexType></element>
  </sequence>
</complexType>

<complexType name="crtType">
  <sequence>
    <element name="crtDialect" type="Q2:crtDialectType"
maxOccurs="1" minOccurs="1">
</element>
    <element name="comment" type="Q3:contentType" maxOccurs="1"
minOccurs="1">
</element>
    <element name="visualRepresentation" type="string"></element>
    <element name="crtSockets" type="string"></element>
    <element name="adaptationBehaviour" type="string"></element>
    <element name="constraints" type="string"></element>
    <element name="associatedDMRelations" type="string"></element>
  </sequence>
</complexType>
</schema>

```

3.7 CAM external language

The CAM external language is the portable format for the output for the Grapple Adaptation Tool (GAT). In essence it is a scaled down version of the CAM-internal language, with only the information relevant for delivery. Therefore it does not have any information for visualising the CAM model, but it does have all other information.

Below we illustrate the new CAM External language via the same example use in section 3.5. We see this type of CAM output as XML descriptions of groups of concepts and *named typed* relations between them. For example, if concept A is to be seen before concept B, a CAM external would be:

```

<model>
  <header> .. </header>
  <body>
    <cam>
      <camInternal>
        <domainModel>..</domainModel>
        <crtModel> .. </crtModel>
        <crt>
          <uuid>cf5de7f5-12b5-4720-92e5-736cac59985b</uuid>
          <camSocket>
            <uuid>e9b45bd0-6013-11de-8a39-0800200c9a66</uuid>
            <socketId>cf5de7f5-12b5-4720-92e5-zzzzzzzzzz</socketId>
            <entity><dmId>201-de-8a39-0800200c9a66</dmId></entity>
          </camSocket>
          <camSocket>
            <caption>target</caption>
            <uuid>f539bae0-6013-11de-8a39-0800200c9a66</uuid>
            <socketId>2b5-4720-92e5-pppppppppppp</socketId>

```

```

    <entity><dmId>11de-8a39-0800200c9a66</dmId></entity>
  </camSocket>
</crt>
</camInternal>
</cam>
</body>
</model>

```

4 Implementation and Integration of the CAM Tool

The basic design of the CAM Tool has already been outlined in D3.3a. It has been described that the tool has a user interface where the author can create, edit and modify CAMs. CAMs should be stored on the server side using Web service and a database behind the Web service.

For the reason of a more efficient implementation the common functionalities of DM Tool, CRT Tool, and CAM Tool have been integrated into the GRAPPLE Authoring Tools (GAT) shell. The functionalities of the GAT shell are provided to all of these tools. The most important functionalities are providing a container including a menu bar and handling the creation and updating of data models.

4.1 The GAT shell and Web Service

The different models (DM, CRT and CAM) each have an authoring tool to author the specific models. The goal of the GAT is ultimately to make it possible for authors (teachers) to create adaptive courses. In order to present a coherent approach and not to frustrate the authors with an array of different tools it is very important that the GAT offers the tools under one umbrella. In the implementation we have decided to take this one step further and to create one completely integrated tool. For an author there is now only one tool, the GAT, which offers DM, CRT and CAM editing functionalities. The fact that the models and tools have been developed in parallel by different teams is completely transparent to the user.

The file *Settings.xml* contains the address of the web services used by the GAT, as well as the names of the models; i.e. crt, cam, dm etc, which will appear as options for creating new models as well as for opening models. In the future new models can be added by adding their name here, and making sure that this name is used in their modeltype tag in the header as well as that there is a folder of the appropriate name in the modules folder in the trunk and a <<name>>.mxml file in that folder containing the model. A new model should extend the same class and react to the same events as other models.

A guide for tool implementers for integrating their tool with the GAT shell can be found in Appendix 8.2. The shell uses the *dockable* Flex library and the *Birdseye ravis* library. The main functionality it offers is a fully functional toolbar. Implementing tools are expected to extend *closablevbox* and react to events.

4.2 Implementation of the CAM Tool

The CAM tool was implemented, as designed in Deliverable 3.3a. 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.

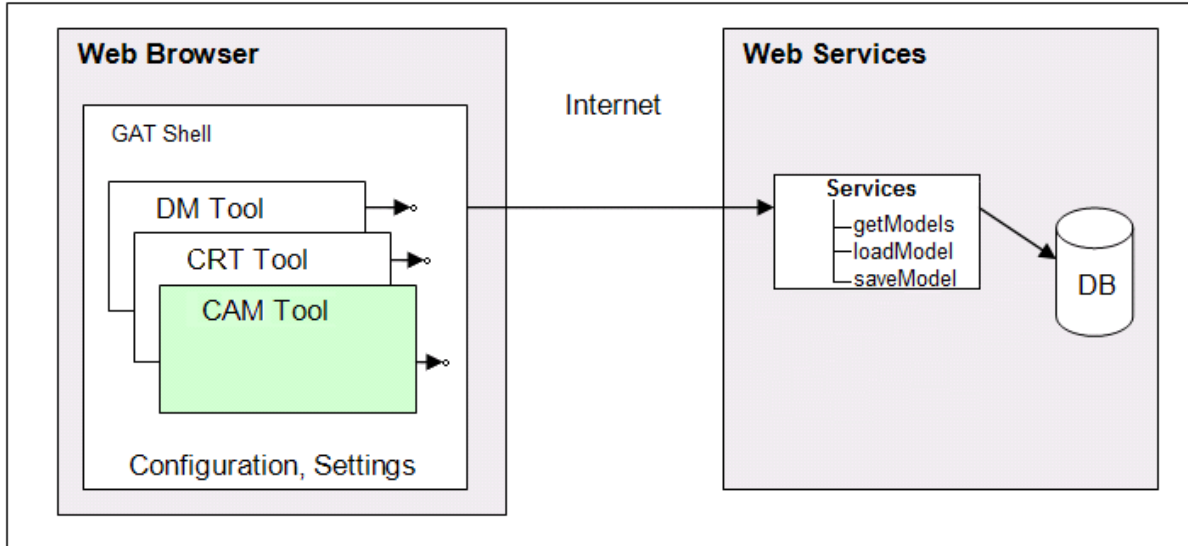


Figure 4. GAT Tool Architecture

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 models (CAMs, DMs and CRTs) a library. It is worth noting that while there is only one webservice, which handles the different models transparently. For this reason, all models have a common header, see section 3.2. The CAM tool, the DM tool and CRT tool use the generic functionalities in the GAT shell to communicate with the web service.

4.3 Integration of the CAM Tool with the GAT shell and Web Service

The GAT shell is a Flex library which can be integrated into the tools. It provides a graphical container, a menu and the functionality for handling models. Figure 5 depicts how the CAM Tool is integrated with the GAT shell.

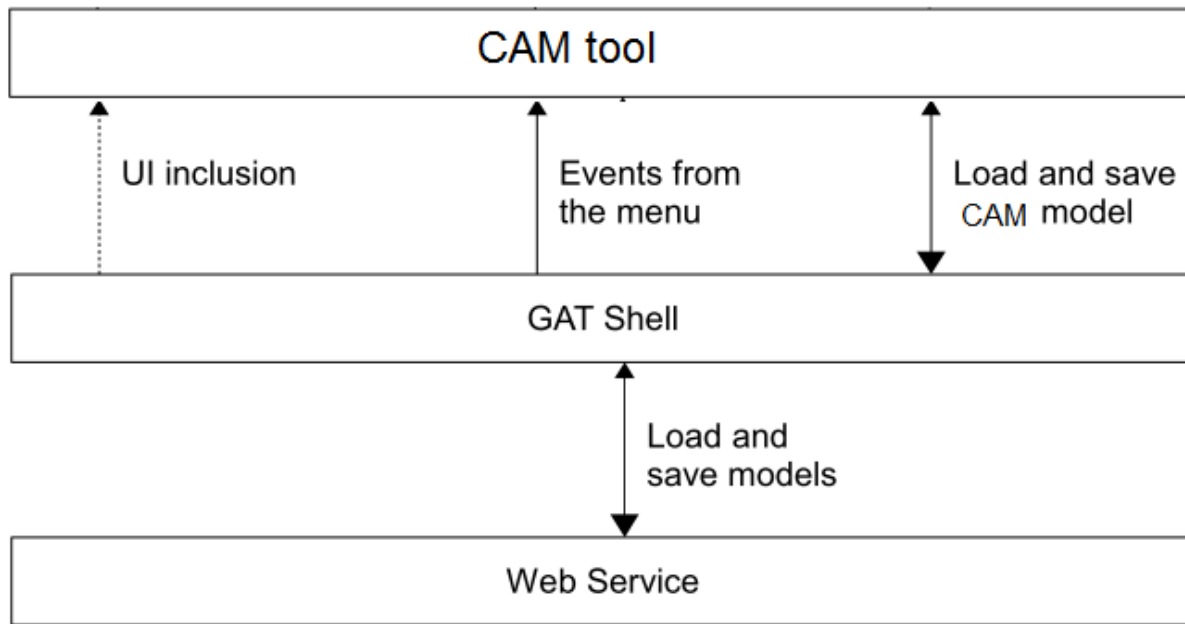


Figure 5: The CAM Tool integrated in the GAT shell

Since the GAT shell provides the menu the events raised if a user activates a menu, items have to be sent to the CRT Tool. The most important events are "open model", "save model", and "save as". These events are received by the application logic which activates the transformation of the internal CRT representation to and from XML format. The XML code will be sent to or retrieved from the GAT shell. The GAT shell is responsible for saving and loading the models to and from the Web service.

5 Preliminary evaluation

The evaluation was conducted in two ways. First of all, informal feedback was gathered from attendees of the ECTEL 2009 conference at a demo session. In addition to this, closed and open ended questions were asked in our questionnaires. Contacts gathered at the ECTEL 2009 and ICALT 2009 conference as well as PhD students in the Department of Computer Science at the University of Warwick, were kindly asked to fill out these. The authoring tools were evaluated using both qualitative and quantitative methods.

5.1.1 Qualitative evaluation

From informal discussions at the demo session at the ECTEL 2009 conference, we got the following feedback. Our 'hypotheses' for the informal and formal qualitative feedback was:

H_{qual} . Users are able to understand how the system works, find the interface usable and agree with the ideas behind the tools.

5.1.1.1 Feedback concerning the GAT/shell in general:

- Escape should also exit the selection mode alongside CTRL+L.
- It would be nice to be able to put a DM and CAM side by side and drag & drop concepts across.
- The functionality the right click on a selection and press copy is desirable.
- The naming of the tools seems confusing; many people asked what is 'DM, CRT CAM'? As an explanation, domain, relation and course seemed to be understood by most.

5.1.1.2 Feedback concerning the DM-tool:

- When selecting some nodes some times turn red, without apparent reason.
- The functionality to right click on a concept and press copy, such as in the CAM is desirable.
- Importing domains into the DM tool would be beneficial, for example from ontologies or IMS-LD?
- When you click a concept and go to insert relation, the concept clicked should be selected as source, it would also be nice to have it in the functionality under the right mouse button.
- It would be nice to be able to click on a concepts name to edit it.

5.1.1.3 Feedback concerning the CRT-tool:

- The naming of the options 'public for LMS' and 'Public for GRAPPLE' in the UM variables part is confusing, a better naming convention has to be found.
- The list of sockets sometimes displays XML rather than the names of the socket.
- The title of a model does not appear in the tab.

5.1.1.4 Feedback concerning the CAM-tool:

- All links appear directed, even the undirected CRTs.
- There may be issues with the scalability of the CAM view, when the course becomes large the view becomes cluttered. Introducing hierarchies or levels was suggested.

We also gathered quantitative feedback in a slightly more formal way, using open ended questions in our questionnaire. The questionnaire was filled in by 10 of the people we contacted. Below we give a summary of the answers they gave to our open ended questions:

UFQQ1: What did you like best about the system and authoring tools?

Answer	Frequency	Answer Statements
Versatility	1	*Granularity and versatility of the system
Possibility of reintegration of components for different purpose	1	*It seems that the various components can be reintegrated to do a slightly different thing (e.g. educational game)
Usability/interface	5	*Usability is ok if you have been trained to use the system *The interface was nice and very easy to use. *Graphical display *The user interface; the navigation bar *The layout of the user interface
Help	2	*The user help *The user manual, it's very helpful

UFQQ2: What did you like least about the system and authoring tools?

Answer	Frequency	Answer Statements
Possible problems with XHTML	1	*generating XHTML with the Grapple specific tasks might be errorprone. Either a short-hand syntax or a graphical IDE with code hints would need to be used.
Complexity	2	*Pages can get very complex *Lots of fields to fill in!
No testing capabilities	1	*For quality assurance the page needs to be tested somehow but I did not see any debugging capabilities for that. So I would need to see somehow what variables are used in the page, and have a possibility to manually enter those
Association of resources in DM	1	*As far as I noticed, the concepts (ontology) directly maps to a page of information. I did not see the possibility to associate "fly-by" topics, that is, some topics that are done in various other pages and will contribute to the knowledge of an item - not specifically on "its" page.
No complete example	1	*If there is a complete example to use this system then it will rather be easy as compare to this tutorial.
Menu – tabs	1	*Two level tab menu
CRT concept	1	*The concept of the CRT is difficult to understand, even with the user manual.

UFQQ3: What should be improved and how?

Answer	Frequency	Answer Statements
Allow user contributions/annotations	1	*Learning in an informal process most likely involves other peoples as well. A clear support for that (user contributed annotations to pages) would be important, I think.
Create possibilities for collaboration	1	*Building up knowledge in a team might be an interesting topic as well. In that case, one would have a "user data model" and a "collective data model" - the user data model would have the individual knowledge status, the collective data model the model of the group. That could lead to interesting learning contexts, ranging from possibility to display "someone in your group already knows that, you try to ask him for more information", or to more sophisticated scenarios, where a competitive learning style

		could be used.
Add tooltips	1	*Tooltips could be added to buttons, windows, tabs, and so on, to aid the user in navigation and use of the system
Provide example	2	*Example should be given in such a way that any body who uses this tutorial can understand how we can implement in real learning environment. *It will be helpful if a demo adaptive material is given first and then the steps to create the material is given behind
Documentation	1	*Documentation

5.1.1.5 Qualitative evaluation conclusions

The qualitative evaluation results partially support hypotheses H_{qual} . Users are mostly able to understand the system. However, the naming of DM-CRT-CAM is confusing and can be improved. (We propose domain – relationship – lesson). Also people are missing real examples. This is probably due to the fact that there is no real course available; at the moment of evaluating there was only a toy-example, and the fact that the conversion was not yet in place, hence authors could see their courses in action.

Respondents seem to mainly agree with the ideas; useful suggestions include support for collaboration, and different roles (people), as well as various small interface improvements. Still some people hold misconceptions regarding the mapping between concepts and pages (this does not have to be one to one, but the current examples only show courses where there is a one to one relation).

5.1.2 Quantitative evaluation

For the formal quantitative evaluation we used the closed questions on the questionnaire. In addition to the open ended questions discussed previously. For the closed questions we had 10 responses. The questionnaire existed of 12 closed questions. The first 10 questions were the questions of the standard SUS (System Usability Scale, [1]) questionnaire for system usability. In addition to this we added the following two closed questions:

UA1 I would like to use this system in the future.

UA2 I would recommend this system to my colleagues.

For the quantitative evaluation we have the following hypotheses:

H_{sus1} Respondents would like to use the tool frequently

H_{sus2} The tool is not perceived as (too) complex

H_{sus3} The tool is easy to learn

H_{sus4} Respondents do not require a lot of support

H_{sus5} The tool is well integrated

H_{sus6} The tool is not inconsistent

H_{sus7} Respondents learn to use the tool quickly

H_{sus8} The tool is not cumbersome

H_{sus9} Respondents are confident to use the tool

H_{sus10} Respondents do not need to learn a lot to use the tool

H_{UA1} Respondents would like to use this system in the future

H_{UA2} Respondents would recommend this system to their colleagues

The (average) results of the SUS questions are shown below. The perfect score would show a perfect star shape. What we can see from these results, is that respondents were not very confident in using the system, probably caused by the fact that they felt they had to learn a lot and didn't learn as quickly as they would have liked. The respondents also felt that they needed support. This could be to do with the lack of good examples, which was highlighted in the qualitative evaluation. In addition to the existing toy example, a real (small) course as example would be very helpful, probably along with a working conversion, so authors can see the resulting course in action. Respondents do feel the system is well integrated and consistent.

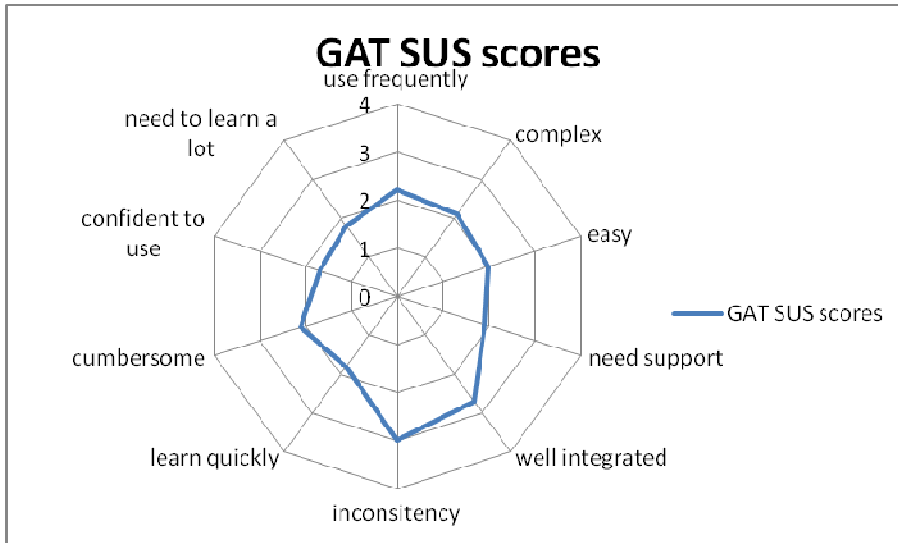


Figure 6 SUS score GAT

In order for our hypotheses to be confirmed, the mean of the related closed question needs to be above 2 on a 0 to 4 scale. An additional requirement, for our hypotheses to be confirmed, is that this could not have been caused by chance. To verify this we use a one-sample t-test against an expected population mean of 2. The standard null hypothesis for this test is 'the mean in our data is equal to 2' and the alternative hypotheses is 'the mean in our data is not equal to 2'. This test works under a number of assumptions:

- The observations are independent from one another.
 - Our evaluation setup, consisted in contacting people directly with the web-address of the tool, user guide and questionnaire. It is therefore reasonable to assume that the observations are independent.
- The sample was drawn from a normal population, however small departures from normality do not affect the test.
 - It is reasonable to assume our data fits these criteria.

We assumed a confidence of 95%. Thus, a hypothesis can be confirmed, if the mean is $M > 2$ and the probability is $P < 0.05$. As we can see in the table below the individual means seem to support the suggestions made by the SUS graph, and it seems like respondents would like to use the system again and recommend it, however due to small sample size the data is not statistically significant.

Table 1 GAT hypotheses results

Hypothesis	T	Df (10 respondents)	Mean difference	P
H _{sus1}	0.802	9	0.200	0.443

H _{SUS2}	0.318	9	0.100	0.758
H _{SUS3}	0.000	9	0.000	1.000
H _{SUS4}	-0.264	9	-0.100	0.798
H _{SUS5}	3.280	9	0.700	0.010
H _{SUS6}	3.000	9	1.000	0.015
H _{SUS7}	-0.612	9	-0.200	0.555
H _{SUS8}	0.429	9	0.100	0.678
H _{SUS9}	-0.896	9	-0.300	0.394
H _{SUS10}	-0.408	9	-0.200	0.693
H _{UA1}	1.078	9	0.400	0.309
H _{UA2}	1.000	9	0.300	0.343

5.2 Conclusions

In this chapter we have introduced a novel graphical approach to authoring of Adaptive Hypermedia, featuring both a graph-based approach as well as a strict separation between content and adaptation. The CAM model and the GAT tool based on the model, allows non-technical authors to use create a conceptual domain, and to use pre-defined relations, in order to create a course. In the CAM model, the adaptive behaviour is represented in abstract re-usable relationships (CRTS) between groups of concepts. In this chapter we have shown the languages used to express the parts of the CAM model, and we have introduced the design of the GAT tool based upon the model.

Finally we have evaluated the tool. From this evaluation we have seen that authors are mostly able to understand the system, and seem to like it. However there are a number of small interface issues, as well as the tool seems to have quite a steep learning curve. This can be improved by providing better example courses, and making the conversion to adaptive systems available, so authors can see their courses in action.

6 User Guide on the CAM Tool

The user guide for the CAM tool, is part of the user guide for the integrated GAT. The user guide will be made available to the users via the tool itself. In the menu users can select “*Help->Help*”, which will open the HTML page containing the user guide. The user guide is also part of deliverable D9.2 and will be updated based upon opinions from WP9 collaborators as well as the results of the formal review of deliverable d9.2. As the user guide is integrated into the tool, evaluation may show results regarding its usability, however there is no specific evaluation of the user guide planned. In the scope of WP9, some preliminary tests in the GRAPPLE tutorial at the ECTEL conference in 2009 in Nice have taken place, in order to highlight some of the more prominent bugs and issues.

The Concept Adaptation Model is initially the 3rd panel available to the author when (s) he opens GAT, as displayed in Figure 7. The CAM effectively contains a course. The CAM will consist of all Domain Models, CRT descriptions used and their instantiations.

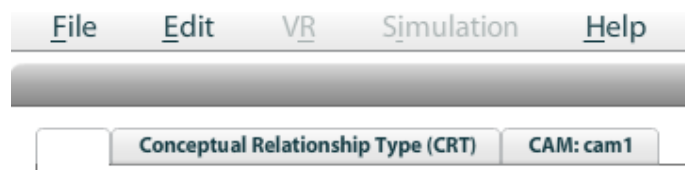


Figure 7. Opening the GAT tool

The author can create a new CAM by selecting “*File->New*” from the menu bar as depicted in Figure 8:

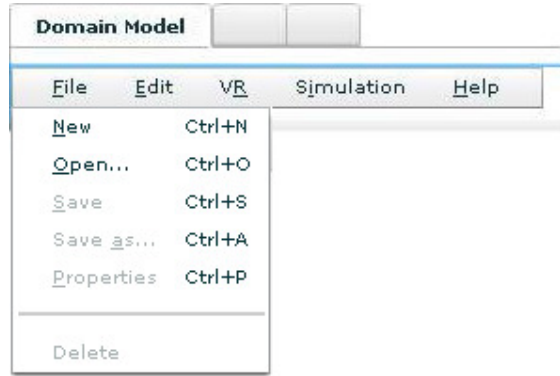


Figure 8. Creating a new CAM

The author has to name the new CAM and then (s) he can start creating the CAM model, by pointing on the CAM editor and right-click. As shown in Figure 9, (s) he can insert a new CRT instance.

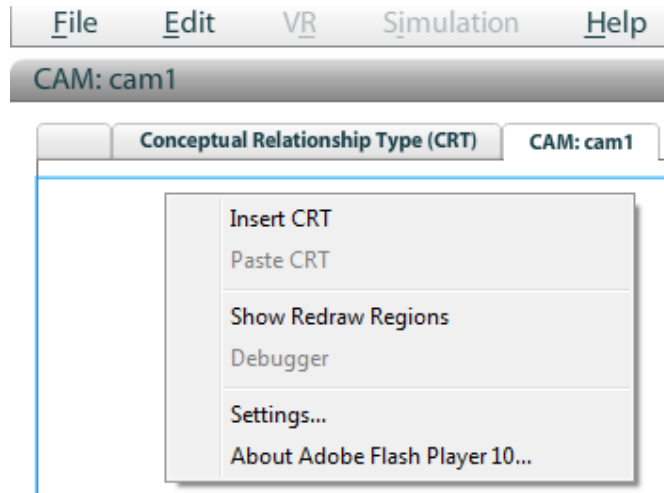


Figure 9. Inserting a CRT instance into a CAM

6.1 Inserting CRTs

A dialog is presented to the author where (s) he can select the CRT model (s) he wishes to insert an instance of. See Figure 10.

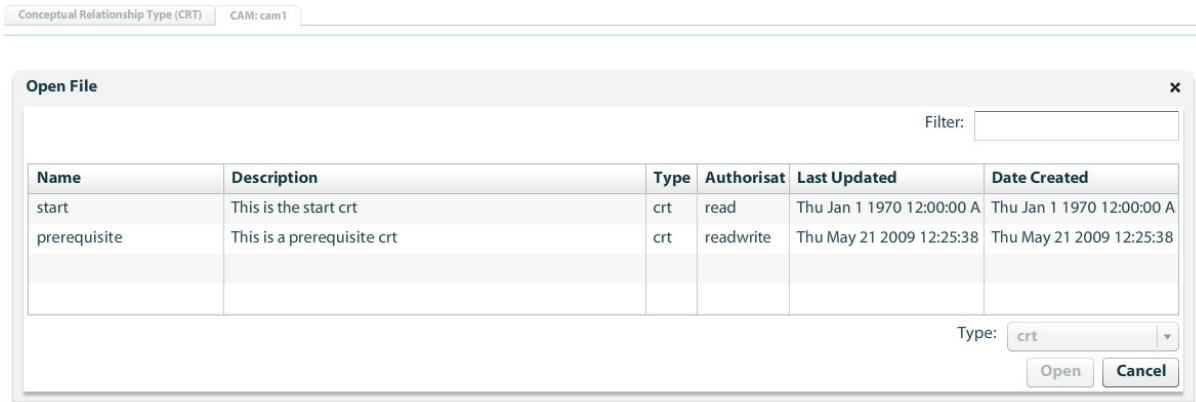


Figure 10. Choosing a CRT to insert

After choosing a CRT for insertion, a CRT node with its sockets is inserted. Initially the sockets do not contain any concepts. See Figure 11.

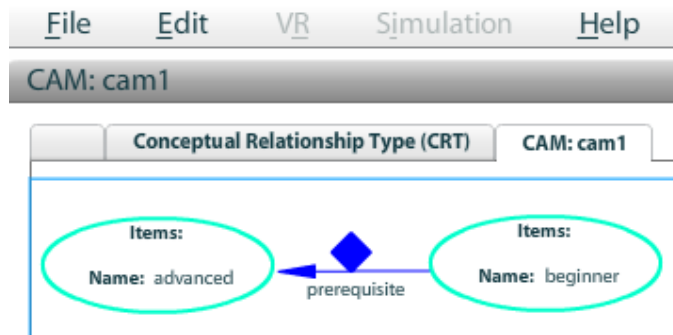


Figure 11. The prerequisite CRT, which has 2 sockets, inserted into the CAM

As an alternative, CRTs can also be copied and pasted. Copying a CRT is done by right-clicking on a CRT node and selecting “Copy CRT”. Then, pasting a CRT is done by right-clicking on an empty space on the canvas and choosing “Paste CRT”. This pastes the CRT instance, including modified colours and shapes, but the sockets will not contain any concepts. See Figure 12 and Figure 13. Please note that CRT instances can be copied from other CAM windows as well as the current window.

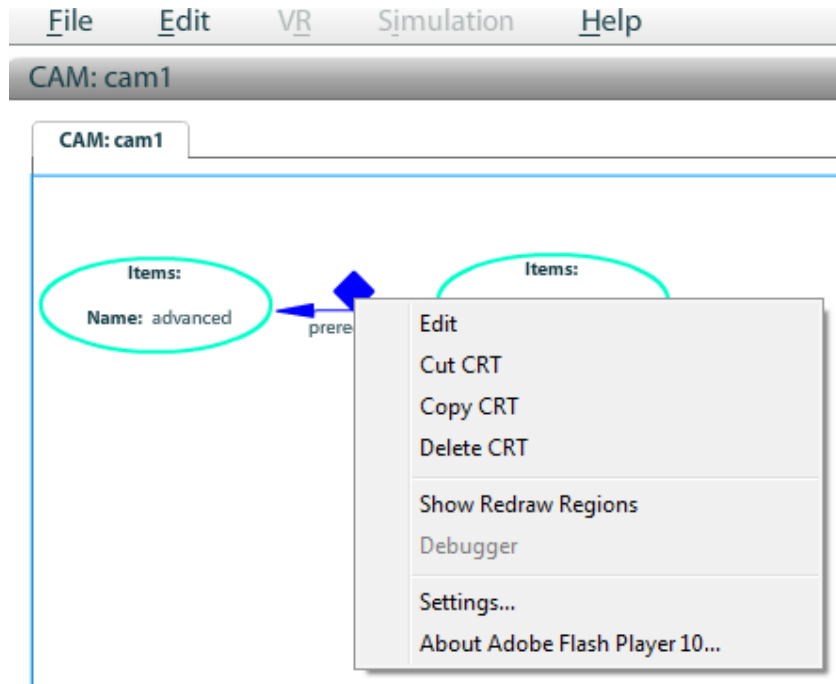


Figure 12. Copy a CRT instance. Other actions are delete and cut (copy then delete)

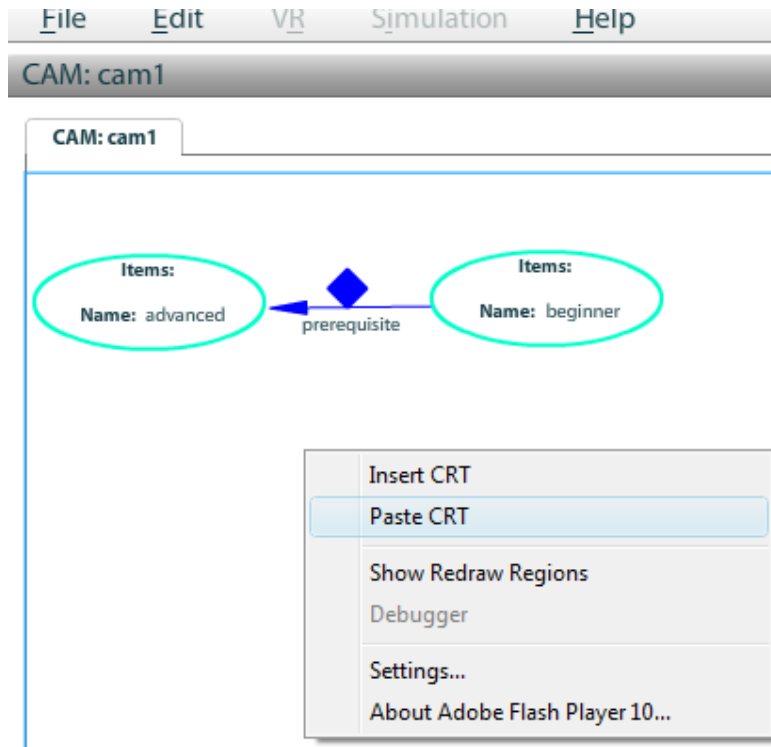


Figure 13. Past CRT is now enabled

6.2 Filling sockets with concepts

Now the author has to fill all sockets with at least one concept. There are various ways of doing this:

- Copy-paste concepts from a CAM window (note: this can be a different CAM window from the current one) see Figure 14, Figure 15 and Figure 16.
- Copy-paste concepts from a Domain Model window (see section 6.2, Figure 15 and Figure 16).
- Through the edit socket interface (see Figure 16, Figure 17, and Figure 18). This also allows insertion of external locations (see Figure 19).

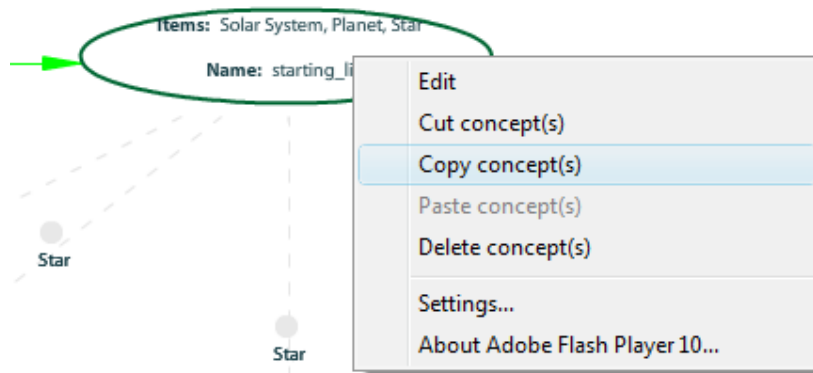


Figure 14. Copy concepts from a socket in a CAM window

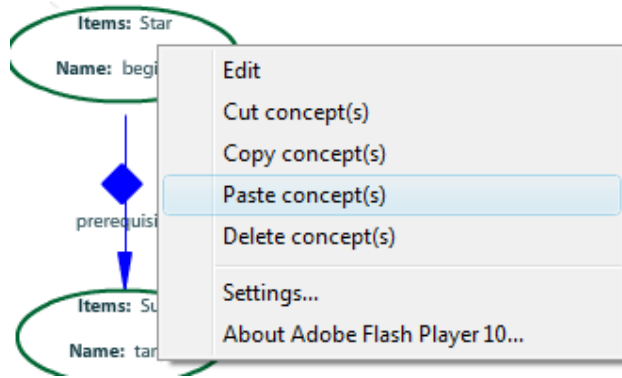


Figure 15. Pasting Concepts into sockets

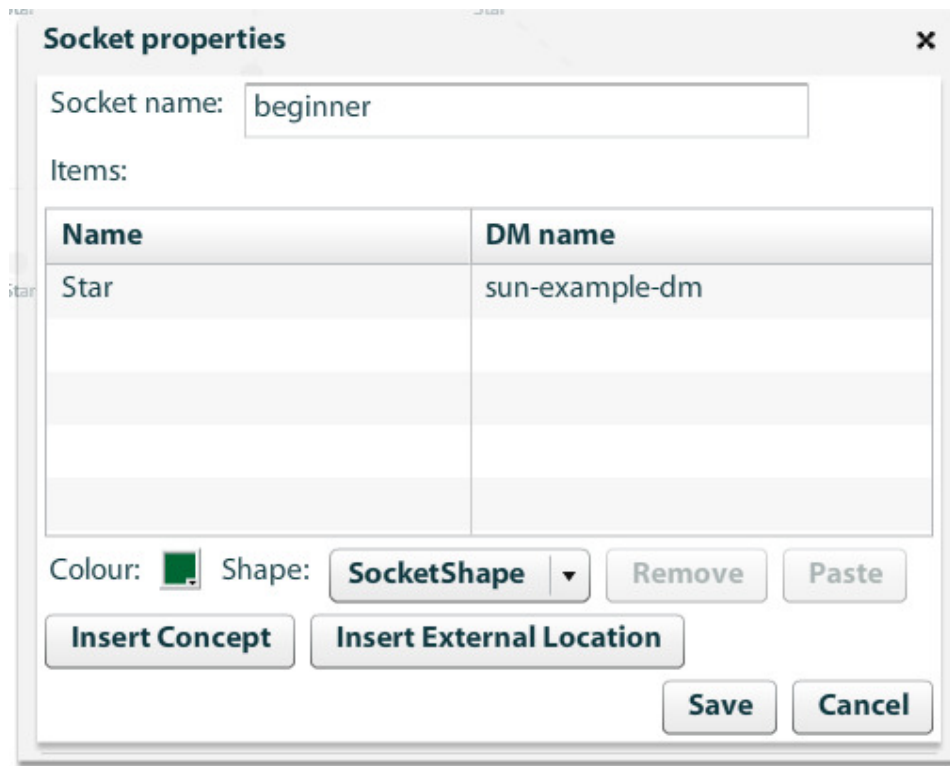


Figure 16. Socket editing screen providing an alternate place to paste concepts and an interface to insert concepts

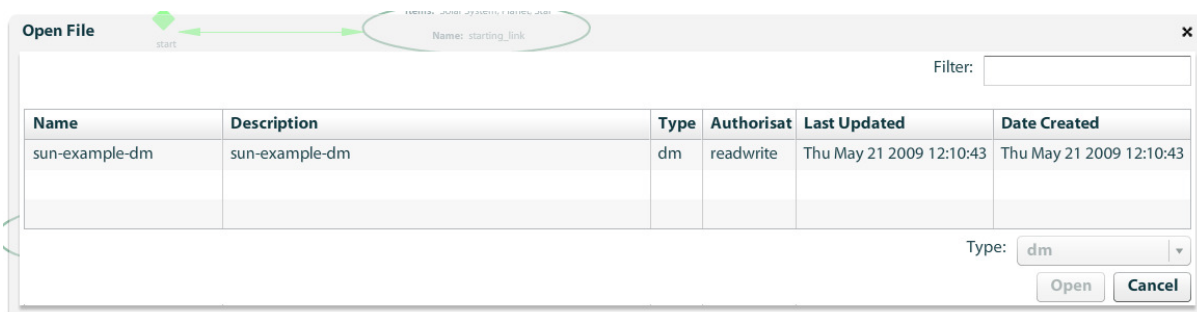


Figure 17. Selection interface for the Domain Model the concept comes from

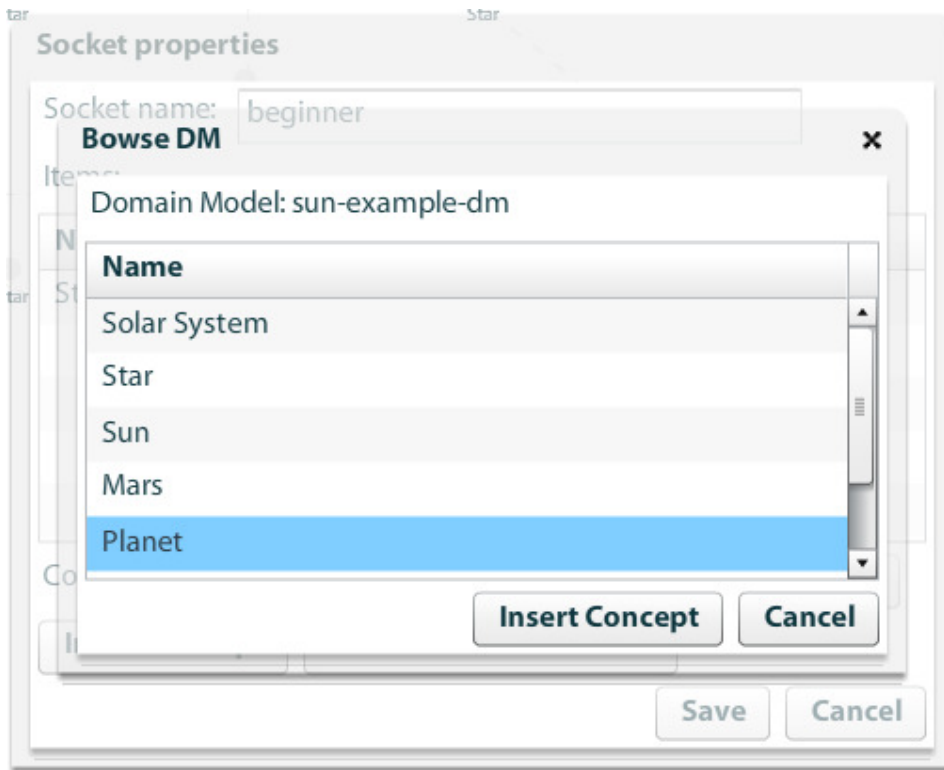


Figure 18. Selection of the Concept

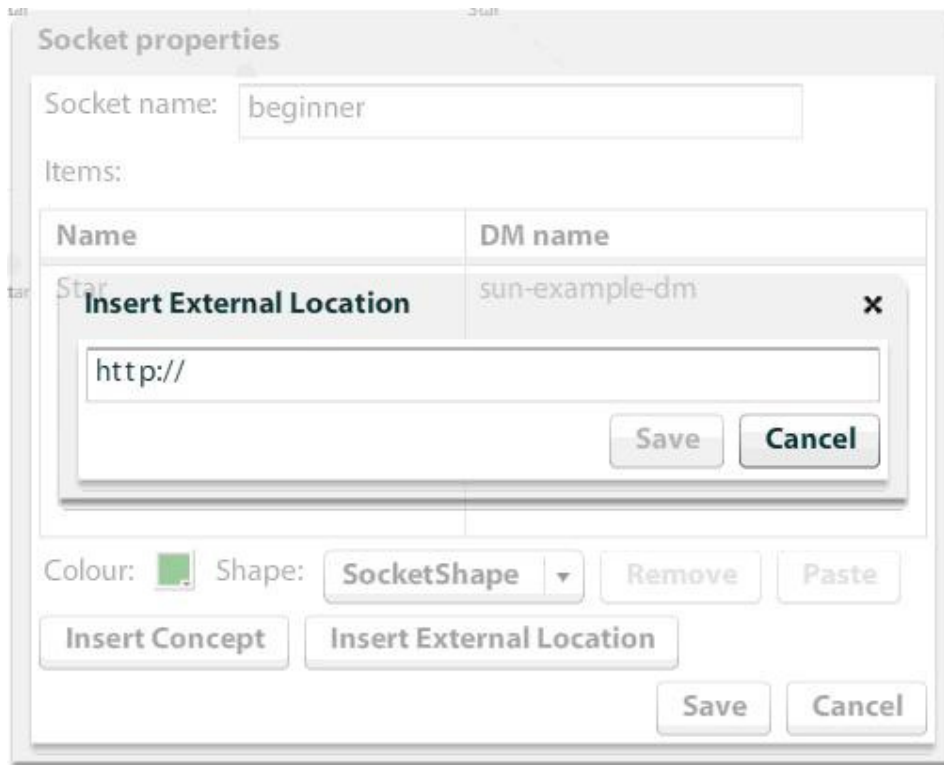


Figure 19. Insert External Location

Please note that the tool will automatically link sockets having some concepts in common in order to indicate this relationship (see Figure 20).

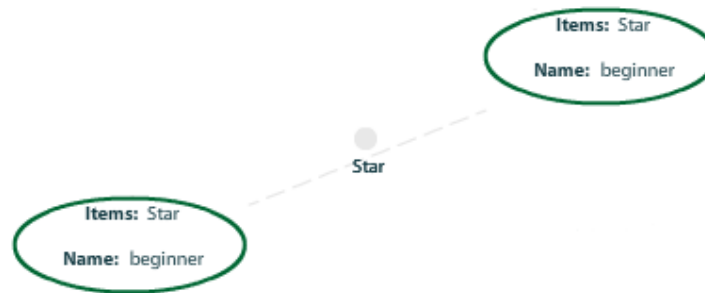


Figure 20. Two sockets containing the concepts Star

6.3 Editing CRTs and Sockets

CRTs and Sockets can be edited by right-clicking on the socket or CRT node and choosing edit or by double clicking.

The screen for editing properties of sockets is shown in Figure 16. It offers the possibility to change the name, shape and colour of the socket. It also allows pasting removing and inserting concepts and external locations.

The screen for editing CRT instances is shown in Figure 21. The dialog allows changing the colour and shape of the CRT node. It also allows launching a new window of the CRT-tool with the description of this CRT instance loaded for editing. Finally it allows updating the CRT description in this model for this type of CRT from a CRT description stored in the repository.

Hence, in order to make changes to the description of the CRT and ultimately to the adaptation code, in this CAM model an author has to open the CRT model in a CRT-tool window. This can either be done by locating it in the menu bar under “File -> Open” or by clicking “Edit CRT” in the CRT editing screen. Then the author has to make the desired changes and store the CRT description – either under the same or a different name. Finally the author has to navigate to an instance of the CRT model that (s) he wanted to change and click “Update CRT”. At this stage the author has to locate the CRT through a dialog which is identical to the one under “file -> open” in the menu bar. This dialog however only lists CRTs that will fit in this place.

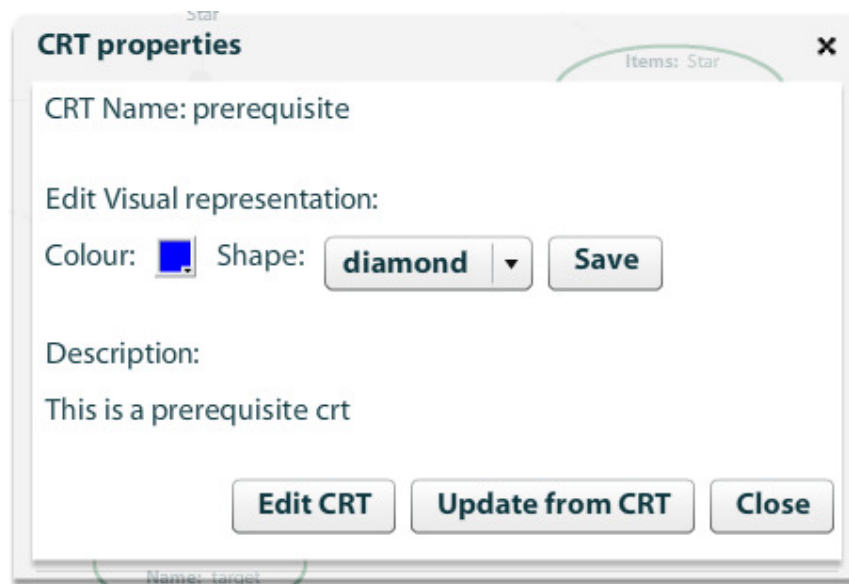


Figure 21. CRT editing screen

6.4 Deleting concepts and CRTs

Concepts can be deleted by right-clicking on a socket and selecting “Delete concepts” (see Figure 22). This will delete all concepts from the socket. Alternatively, in the socket edit dialog, see Figure 16, the author can select any number of concepts and delete only the selected ones by clicking *remove* and then *save*.

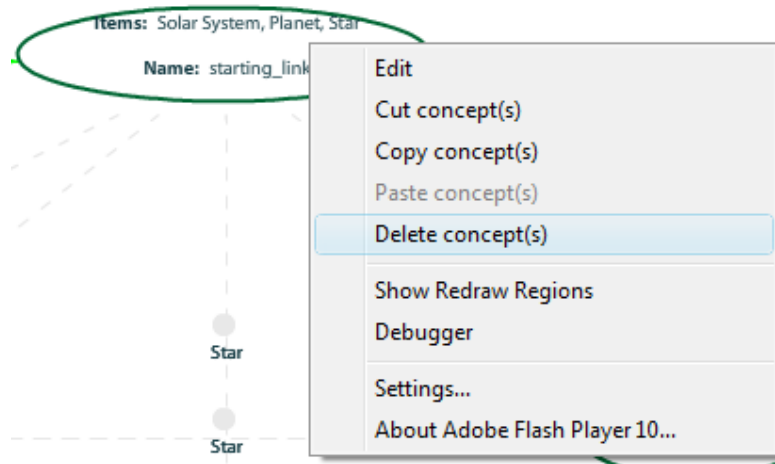


Figure 22. Deleting all concepts from a socket.

7 Conclusion and Outlook

This deliverable describes the initial development of the CAM Tool. The design has been extensively described in Deliverable D3.3a. The initial version of the tool is now working and fulfils the basic requirements identified in D3.3a. CAMs can be created and modified and these CAMs can be loaded and saved in XML format to and from a database on the Web. The tool is Web-based and provides a graphical interface for the author.

The final version of the CAM Tool will be delivered and described in Deliverable D3.2c (due month 30, July 2010). The results of evaluation conducted in wp9 and wp10 will be taken into account and influence the further work on the CAM Tool. The final version will also include a number of bug-fixes, based on bugs gathered both at informal testing by the WP3 partners as well as at the evaluation. For bug tracking the GIDTS tool (<http://www.grapple-project.org/gidts-1>) is used. The final version will also include at least the relationshipType element as described in 3.5. In addition to this an improved copy-paste functionality (copy buttons available under the right mouse button in the DM) and a drag and drop functionality between DM and Cam are planned.

References

1. Brooke, J.: SUS: a "quick and dirty" usability scale. In: Jordan, P.W., Thomas, B., Weerdmeester, B.A. and McClelland, A.L. (eds.). Usability Evaluation in Industry. London: Taylor and Francis. 1996
2. M. Cannataro and A. Pugliese, "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.
3. A.I. Cristea, Adaptive Course Creation for All, ITCC'04, International Conference on Information Technology, Las Vegas, US, IEEE, 2004
4. Cristea, A.I., Calvi, L. The three Layers of Adaptation Granularity, UM'03, Springer.
5. A.I. Cristea, A. de Mooij, 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
6. De Bra, P., D. Smits, D., Stash, N., The Design of AHA! , ACM Conference on Hypertext and Hypermedia, pp. 133, Odense, Denmark, 2006
7. F. Garzotto and A.I. Cristea ADAPT: Major design dimensions for educational adaptive hypermedia. ED-MEDIA 2004 Conference, 2004
8. F. Halasz and M. Schwartz, M. (1994). The Dexter hypertext reference model: Hypermedia. Communications of the ACM, 37(2), 30-39.

9. N. Koch and W. Wirsing, Software Engineering for Adaptive Hypermedia Applications? 3rd Workshop on Adaptive Hypertext and Hypermedia, UM 2004 Conference, 2001
10. Koch N., Wirsing M. Software Engineering for Adaptive Hypermedia Applications. 8th International Conference on User Modeling, Sonthofen, Germany, 2001.
11. Stash, N., Cristea, A.I., De Bra, P., Adaptation Languages as vehicles of explicit intelligence in Adaptive Hypermedia, In International Journal on Continuing Engineering Education and Life-Long Learning, vol. 17, nr 4/5, pp. 319-336, InderScience, 2007.
12. van der Sluijs, K., Hidders, J., Leonardi, E., Houben G.J.: Generic Adaptation Language for describing Adaptive Hypermedia, Proc of International Workshop on Dynamic and Adaptive Hypertext: Generic Frameworks, Approaches and Techniques (2009)
13. H. Wu, A Reference Architecture for Adaptive Hypermedia Applications, doctoral thesis, Eindhoven University of Technology, The Netherlands, ISBN 90-386-0572-2.

8 Appendix

8.1 CAM Examples

In this section an example of a CAM definition is given. The example is a simple solar system CAM as used in the user guide.

```

<model>
  <header>
    <modeluuid>8328858e-3161-419c-a96a-99863eb7a1c2</modeluuid>
    <modeltype>cam</modeltype>
    <authoruuid>>null</authoruuid>
    <authorisation>readwrite</authorisation>
    <creationtime>1253140881961</creationtime>
    <updatetime>1254175085904</updatetime>
    <title>Solar System</title>
    <description>test</description>
  </header>
  <body>
    <cam>
      <camInternal>
        <crt type="crt">
          <camCrtUuid>B99C78C6-35E7-FEDE-9D0F-C506EB575985</camCrtUuid>
          <uuid>aaaaaaaa-12b5-4720-92e5-736cac59985b</uuid>
          <shape>diamond</shape>
          <colour>#00FF00</colour>
          <position>
            <x>672.5</x>
            <y>109.5</y>
          </position>
          <camSocket>
            <uuid>0233C4AF-5414-2BEA-E41A-C506EB574C46</uuid>
            <socketid>cf5de7f5-12b5-4720-92e5-qqqqqqqqqq</socketid>
            <colour>26163</colour>
            <shape>SocketShape</shape>
            <position>
              <x>475.675</x>
              <y>109.5</y>
            </position>
            <entity>
              <dmId>989cc040-45f4-11de-8a39-0800200c9a66</dmId>
            </entity>
            <entity>
              <dmId>af56f6f0-4600-11de-8a39-0800200c9a66</dmId>
            </entity>
            <entity>
              <dmId>44c31200-4601-11de-8a39-0800200c9a66</dmId>
            </entity>
            <entity>
              <dmId>a5d15f80-4600-11de-8a39-0800200c9a66</dmId>
            </entity>
          </camSocket>
        </crt>
        <crt type="crt">
          <camCrtUuid>A9F70CA4-1010-87AE-D2BE-C5071FEDD659</camCrtUuid>
          <uuid>cf5de7f5-12b5-4720-92e5-736cac59985b</uuid>
          <shape>Diamond</shape>
          <colour>#0000FF</colour>
          <position>
            <x>280</x>
            <y>169</y>
          </position>
          <camSocket>
            <uuid>5D85FA82-B2F5-7C80-A031-C5071FEE9809</uuid>
            <socketid>cf5de7f5-12b5-4720-92e5-zzzzzzzzzzz</socketid>
            <colour>26163</colour>
            <shape>SocketShape</shape>
            <position>
              <x>105.325</x>
              <y>227.5</y>
            </position>
            <entity>
              <dmId>44c31200-4601-11de-8a39-0800200c9a66</dmId>
            </entity>
          </camSocket>
        </crt>
      </camInternal>
    </cam>
  </body>
</model>

```

```

    </entity>
  </camSocket>
<camSocket>
  <uuid>F16898C3-C4F8-BD81-5E66-C5071FEE6252</uuid>
  <socketid>cf5de7f5-12b5-4720-92e5-pppppppppp</socketid>
  <colour>26163</colour>
  <shape>SocketShape</shape>
  <position>
    <x>104.325</x>
    <y>411.5</y>
  </position>
  <entity>
    <dmId>e2506f00-4600-11de-8a39-0800200c9a66</dmId>
  </entity>
</camSocket>
</crt>
<crt type="crt">
  <camCrtUuid>90B2D9EA-1344-3CA0-3F33-C5072C702806</camCrtUuid>
  <uuid>cf5de7f5-12b5-4720-92e5-736cac59985b</uuid>
  <shape>Diamond</shape>
  <colour>#0000FF</colour>
  <position>
    <x>263</x>
    <y>343</y>
  </position>
  <camSocket>
    <uuid>3D06AE6B-183C-BA4C-8DEE-C5072C71A6E5</uuid>
    <socketid>cf5de7f5-12b5-4720-92e5-zzzzzzzzzzz</socketid>
    <colour>26163</colour>
    <shape>SocketShape</shape>
    <position>
      <x>251.325</x>
      <y>224.5</y>
    </position>
    <entity>
      <dmId>a5d15f80-4600-11de-8a39-0800200c9a66</dmId>
    </entity>
  </camSocket>
<camSocket>
  <uuid>6EEB1AD4-7BDB-9F8B-F35F-C5072C71C856</uuid>
  <socketid>cf5de7f5-12b5-4720-92e5-pppppppppp</socketid>
  <colour>26163</colour>
  <shape>SocketShape</shape>
  <position>
    <x>250.625</x>
    <y>415</y>
  </position>
  <entity>
    <dmId>af56f6f0-4600-11de-8a39-0800200c9a66</dmId>
  </entity>
</camSocket>
</crt>
<crt type="crt">
  <camCrtUuid>BA474EAE-5FBF-E193-10C2-C50770E82365</camCrtUuid>
  <uuid>cf5de7f5-12b5-4720-92e5-736cac59985b</uuid>
  <shape>Diamond</shape>
  <colour>#0000FF</colour>
  <position>
    <x>389</x>
    <y>297</y>
  </position>
  <camSocket>
    <uuid>28283A29-2768-B4C5-2FAC-C50770E9447F</uuid>
    <socketid>cf5de7f5-12b5-4720-92e5-zzzzzzzzzzz</socketid>
    <colour>0</colour>
    <shape>SocketShape</shape>
    <position>
      <x>417.125</x>
      <y>329.5</y>
    </position>
    <entity>
      <dmId>44c31200-4601-11de-8a39-0800200c9a66</dmId>
    </entity>
  </camSocket>
<camSocket>
  <uuid>4D9D26F2-8880-2700-FE1B-C50770E9AC99</uuid>
  <socketid>cf5de7f5-12b5-4720-92e5-pppppppppp</socketid>
  <colour>16776960</colour>
  <shape>SocketShape</shape>

```

```

    <position>
      <x>420.325</x>
      <y>451.5</y>
    </position>
  </entity>
  <dmId>cbb6e940-4600-11de-8a39-0800200c9a66</dmId>
</entity>
</camSocket>
</crt>
<crt type="crt">
  <camCrtUuid>626405B2-9DA5-21FF-954A-C50799DBE7F4</camCrtUuid>
  <uuid>cf5de7f5-12b5-4720-92e5-736cac59985b</uuid>
  <shape>Diamond</shape>
  <colour>#0000FF</colour>
  <position>
    <x>539</x>
    <y>242</y>
  </position>
  <camSocket>
    <uuid>EE799E90-4EC4-65EC-533F-C50799DB699A</uuid>
    <socketid>cf5de7f5-12b5-4720-92e5-zzzzzzzzzzzz</socketid>
    <colour>10027008</colour>
    <shape>SocketShape</shape>
    <position>
      <x>567.325</x>
      <y>199.5</y>
    </position>
  </entity>
  <dmId>44c31200-4601-11de-8a39-0800200c9a66</dmId>
</entity>
</camSocket>
<camSocket>
  <uuid>2F0368AC-3992-0621-D3AB-C50799DB032B</uuid>
  <socketid>cf5de7f5-12b5-4720-92e5-pppppppppppp</socketid>
  <colour>3407616</colour>
  <shape>SocketShape</shape>
  <position>
    <x>570.275</x>
    <y>393.5</y>
  </position>
  <entity>
    <dmId>d45f8e30-4600-11de-8a39-0800200c9a66</dmId>
  </entity>
</camSocket>
</crt>
<crtModel>
  <model>
    <header>
      <title>start</title>
      <description>Start CRT</description>
      <creationtime>1253890206022</creationtime>
      <updatetime/>
      <modeluuid>aaaaaaaa-12b5-4720-92e5-736cac59985b</modeluuid>
      <authoruid/>
      <authorisation>readwrite</authorisation>
      <modeltype>crt</modeltype>
    </header>
    <body>
      <crt>
        <comment/>
        <crtdialect>CRT</crtdialect>
        <visualrepresentation>
          <colour>0x00FFCC</colour>
          <shape>diamond</shape>
        </visualrepresentation>
        <adaptationbehaviour>
          <galcode/>
          <usermodel/>
        </adaptationbehaviour>
        <constraints>
          <loopsallowed>false</loopsallowed>
        </constraints>
        <crtsockets>
          <socket type="start_link">
            <uuid>cf5de7f5-12b5-4720-92e5-qqqqqqqqqq</uuid>
            <name>start_socket</name>
            <mincardinality>1</mincardinality>
            <maxcardinality>*</maxcardinality>
          </socket>
        </crtsockets>
      </crt>
    </body>
  </model>
</crtModel>

```

```

        </crtsockets>
        <associateddmrelations/>
    </crt>
</body>
</model>
<model>
  <header>
    <title>prereq-test-mh</title>
    <description>prereq-test-mh</description>
    <creationtime>1253890007331</creationtime>
    <updatetime/>
    <modeluuid>cf5de7f5-12b5-4720-92e5-736cac59985b</modeluuid>
    <authoruuid/>
    <authorisation>readwrite</authorisation>
    <modeltype>crt</modeltype>
  </header>
  <body>
    <crt>
      <comment/>
      <crt dialect>CRT</crt dialect>
      <visualrepresentation>
        <colour>#000000</colour>
        <shape>diamond</shape>
      </visualrepresentation>
      <adaptationbehaviour>
        <galcode/>
        <usermodel/>
      </adaptationbehaviour>
      <constraints>
        <loopsallowed>>false</loopsallowed>
      </constraints>
      <crtsockets>
        <socket type="source">
          <uuid>cf5de7f5-12b5-4720-92e5-zzzzzzzzzzzz</uuid>
          <name>beginner</name>
          <mincardinality>1</mincardinality>
          <maxcardinality>*</maxcardinality>
        </socket>
        <socket type="target">
          <uuid>cf5de7f5-12b5-4720-92e5-pppppppppppp</uuid>
          <name>advanced</name>
          <mincardinality>1</mincardinality>
          <maxcardinality>*</maxcardinality>
        </socket>
      </crtsockets>
      <associateddmrelations/>
    </crt>
  </body>
</model>
</crtModel>
<domainModel>
  <model>
    <header>
      <modeluuid>660e8400-e29b-41d4-a716-446655440000</modeluuid>
      <modeltype>dm</modeltype>
      <authoruuid>null</authoruuid>
      <authorisation>readwrite</authorisation>
      <creationtime>1242904243000</creationtime>
      <updatetime>1242904243000</updatetime>
      <title>sun-example-dm</title>
      <description>sun-example-dm</description>
    </header>
    <body>
      <dm>
        <vdex>
          <term>
            <termIdentifier>989cc040-45f4-11de-8a39-0800200c9a66</termIdentifier>
            <caption>
              <langstring language="it">Solar System</langstring>
            </caption>
            <description>
              <langstring language="en">blabla</langstring>
            </description>
            <mediaDescriptor>
              <mediaLocator>http://host/solarSystem.jpg</mediaLocator>
              <interpretationNote>
                <langstring language="en">some note on the media</langstring>
              </interpretationNote>
            </mediaDescriptor>
          </term>
        </vdex>
      </dm>
    </body>
  </model>
</domainModel>

```

```

<metadata/>
</term>
<term>
  <termIdentifier>a5d15f80-4600-11de-8a39-0800200c9a66</termIdentifier>
  <caption>
    <langstring language="it">Star</langstring>
  </caption>
  <description>
    <langstring language="en">blabla</langstring>
  </description>
  <mediaDescriptor>
    <mediaLocator>http://host/star.pdf</mediaLocator>
    <interpretationNote>
      <langstring language="en">some note on the media</langstring>
    </interpretationNote>
  </mediaDescriptor>
  <mediaDescriptor>
    <mediaLocator>http://en.wikipedia.org/wiki/star</mediaLocator>
    <interpretationNote>
      <langstring language="en">some note on the media</langstring>
    </interpretationNote>
  </mediaDescriptor>
  <metadata/>
  <index>0</index>
</term>
<term>
  <termIdentifier>af56f6f0-4600-11de-8a39-0800200c9a66</termIdentifier>
  <caption>
    <langstring language="it">Sun</langstring>
  </caption>
  <description>
    <langstring language="en">blabla</langstring>
  </description>
  <mediaDescriptor>
    <mediaLocator>http://host/sun.x3d</mediaLocator>
    <interpretationNote>
      <langstring language="en">some note on the media</langstring>
    </interpretationNote>
  </mediaDescriptor>
  <mediaDescriptor>
    <mediaLocator>http://en.wikipedia.org/wiki/sun</mediaLocator>
    <interpretationNote>
      <langstring language="en">some note on the media</langstring>
    </interpretationNote>
  </mediaDescriptor>
  <metadata/>
  <index>1</index>
</term>
<term>
  <termIdentifier>cbb6e940-4600-11de-8a39-0800200c9a66</termIdentifier>
  <caption>
    <langstring language="it">Mars</langstring>
  </caption>
  <description>
    <langstring language="en">blabla</langstring>
  </description>
  <mediaDescriptor>
    <mediaLocator>http://en.wikipedia.org/wiki/mars</mediaLocator>
    <interpretationNote>
      <langstring language="en">some note on the media</langstring>
    </interpretationNote>
  </mediaDescriptor>
  <dmId>cbb6e940-4600-11de-8a39-0800200c9a66</dmId>
  <entity>
    <dmId>cbb6e940-4600-11de-8a39-0800200c9a66</dmId>
  </entity>
</term>
<term>
  <termIdentifier>44c31200-4601-11de-8a39-0800200c9a66</termIdentifier>
  <caption>
    <langstring language="it">Planet</langstring>
  </caption>
  <description>
    <langstring language="en">blabla</langstring>
  </description>
  <mediaDescriptor>
    <mediaLocator>http://en.wikipedia.org/wiki/Planet</mediaLocator>
    <interpretationNote>
      <langstring language="en">some note on the media</langstring>

```

```

        </interpretationNote>
    </mediaDescriptor>
</metadata/>
<dmId>44c31200-4601-11de-8a39-0800200c9a66</dmId>
<entity>
    <dmId>44c31200-4601-11de-8a39-0800200c9a66</dmId>
</entity>
</term>
<term>
    <termIdentifier>d45f8e30-4600-11de-8a39-0800200c9a66</termIdentifier>
    <caption>
        <langstring language="it">Earth</langstring>
    </caption>
    <description>
        <langstring language="en">blabla</langstring>
    </description>
    <mediaDescriptor>
        <mediaLocator>http://en.wikipedia.org/wiki/earth</mediaLocator>
        <interpretationNote>
            <langstring language="en">some note on the media</langstring>
        </interpretationNote>
    </mediaDescriptor>
    </metadata/>
<dmId>d45f8e30-4600-11de-8a39-0800200c9a66</dmId>
<entity>
    <dmId>d45f8e30-4600-11de-8a39-0800200c9a66</dmId>
</entity>
</term>
<term>
    <termIdentifier>e2506f00-4600-11de-8a39-0800200c9a66</termIdentifier>
    <caption>
        <langstring language="it">Jupiter</langstring>
    </caption>
    <description>
        <langstring language="en">blabla</langstring>
    </description>
    <mediaDescriptor>
        <mediaLocator>http://en.wikipedia.org/wiki/jupiter</mediaLocator>
        <interpretationNote>
            <langstring language="en">some note on the media</langstring>
        </interpretationNote>
    </mediaDescriptor>
    </metadata/>
<dmId>e2506f00-4600-11de-8a39-0800200c9a66</dmId>
<entity>
    <dmId>e2506f00-4600-11de-8a39-0800200c9a66</dmId>
</entity>
</term>
<relationship>
    <sourceTerm>af56f6f0-4600-11de-8a39-0800200c9a66</sourceTerm>
    <targetTerm>a5d15f80-4600-11de-8a39-0800200c9a66</targetTerm>
    <relationshipType
source="http://www.grapple.org/relations.xml">is_a</relationshipType>
    </metadata/>
</relationship>
<relationship>
    <sourceTerm>e2506f00-4600-11de-8a39-0800200c9a66</sourceTerm>
    <targetTerm>44c31200-4601-11de-8a39-0800200c9a66</targetTerm>
    <relationshipType
source="http://www.grapple.org/relations.xml">is_a</relationshipType>
    </metadata/>
</relationship>
<relationship>
    <sourceTerm>d45f8e30-4600-11de-8a39-0800200c9a66</sourceTerm>
    <targetTerm>44c31200-4601-11de-8a39-0800200c9a66</targetTerm>
    <relationshipType
source="http://www.grapple.org/relations.xml">is_a</relationshipType>
    </metadata/>
</relationship>
<relationship>
    <sourceTerm>cbb6e940-4600-11de-8a39-0800200c9a66</sourceTerm>
    <targetTerm>44c31200-4601-11de-8a39-0800200c9a66</targetTerm>
    <relationshipType source="http://www.grapple.org">is_a</relationshipType>
</relationship>
<relationship>
    <sourceTerm>989cc040-45f4-11de-8a39-0800200c9a66</sourceTerm>
    <targetTerm>af56f6f0-4600-11de-8a39-0800200c9a66</targetTerm>
    <relationshipType
source="http://www.grapple.org/relations.xml">main_star</relationshipType>

```

```

    <metadata/>
  </relationship>
<relationship>
  <sourceTerm>af56f6f0-4600-11de-8a39-0800200c9a66</sourceTerm>
  <targetTerm>a5d15f80-4600-11de-8a39-0800200c9a66</targetTerm>
  <relationshipType
source="http://www.grapple.org/rerelations.xml">is_a</relationshipType>
  <metadata/>
  </relationship>
<relationship>
  <sourceTerm>989cc040-45f4-11de-8a39-0800200c9a66</sourceTerm>
  <targetTerm>af56f6f0-4600-11de-8a39-0800200c9a66</targetTerm>
  <relationshipType
source="http://www.grapple.org/relations.xml">is_composed_of</relationshipType>
  <metadata/>
  </relationship>
<relationship>
  <sourceTerm>989cc040-45f4-11de-8a39-0800200c9a66</sourceTerm>
  <targetTerm>cbb6e940-4600-11de-8a39-0800200c9a66</targetTerm>
  <relationshipType
source="http://www.grapple.org/relations.xml">is_composed_of</relationshipType>
  <metadata/>
  </relationship>
<relationship>
  <sourceTerm>989cc040-45f4-11de-8a39-0800200c9a66</sourceTerm>
  <targetTerm>44c31200-4601-11de-8a39-0800200c9a66</targetTerm>
  <relationshipType
source="http://www.grapple.org/relations.xml">is_composed_of</relationshipType>
  <metadata/>
  </relationship>
<relationship>
  <sourceTerm>989cc040-45f4-11de-8a39-0800200c9a66</sourceTerm>
  <targetTerm>d45f8e30-4600-11de-8a39-0800200c9a66</targetTerm>
  <relationshipType
source="http://www.grapple.org/relations.xml">is_composed_of</relationshipType>
  <metadata/>
  </relationship>
<relationship>
  <sourceTerm>989cc040-45f4-11de-8a39-0800200c9a66</sourceTerm>
  <targetTerm>e2506f00-4600-11de-8a39-0800200c9a66</targetTerm>
  <relationshipType
source="http://www.grapple.org/rerelations.xml">is_composed_of</relationshipType>
  <metadata/>
  </relationship>
</vdex>
</dm>
</body>
</model>
</domainModel>
</camInternal>
</cam>
</body>
</model>

```

8.2 Information for integrating GAT tools into the GAT shell

For the integration with the shell the following issues need to be observed:

1. **If you run your module stand-alone, copy the assets to your source path.**
2. **Please make your tools implement ClosableVBox**
3. **The events (see 3) need to be handled by the individual tools**

In the sourcepath .../shared/main/flex there are the following folders:

8.2.1 assets

If you run your module stand-alone, copy the assets to your source path.

File	Tool	Use
font	all	Used by css in main

hitskin.png	all	Used by main, floating window library
selectionCursor.gif	all	Selection cursor
settings.xml	all	used for identifying the wsdl for web services
shapes.xml	CRT CAM	indicates which shapes for CRTs and sockets are available

8.2.2 net

contains library for floating windows, as used by main. Please make your tools implement ClosableVBox as in the cam tool (cam.mxml):

```
<goozo:ClosableVBox xmlns:mx="http://www.adobe.com/2006/mxml"
  xmlns:shared="org.grapple_project.authoring.ui.*"
  xmlns:ravis="org.un.cava.birdeye.ravis.graphLayout.visual.*"
  xmlns:grapplevisualgraph="org.grapple_project.authoring.*"
  xmlns:goozo="net.goozo.mx.dockable.*" ...
```

To determine what happens just before the window is close, override closeTab, see `override public function closeTab():Boolean` in cam.mxml

8.2.3 org.grapple_authoring_tool.authoring.events

The events are the main method of communication between the shell and the individual tools. The following events have to be handled by the tools, see cam.mxml for examples.

8.2.3.1 onLoadModel

see `protected function load(event :LoadModelEvent):void` in cam.mxml. The model to be loaded is given. It's guaranteed to be of the correct tupe (i.e. the DM tool will only be asked to open a DM)

8.2.3.2 onSaveModel

The author clicked save, or selected a file for save as. See `protected function save(event :SaveAsModelEvent=null) : void` in cam.mxml. The method uses `toolbar.saveModel` to save the model (this function deals with the WS calls).

8.2.3.3 onNewModel

This event is fired, when the author selects file -> new see `protected function newModel(event:NewModelEvent) : void` in cam.mxml. This function will only be called on a newly opened window of a DM/CRT/CAM tool, so you can assume that this instance wasn't already holding any models.

8.2.3.4 onMenuItemClick

The author clicked an item on the menubar. This event fires for all buttons on the menubar, with a switch the relevant actions can be coupled to buttons. See `protected function menuItemClicked(event :MenuItemClickEvent) : void` in cam.mxml

8.2.3.5 onUpdateCurrentModel

With this method the toolbar notifies a dm/crt/cam-tool window that it wants to know the model currently opened in the tool in its current state. `toolbar.currentModel` should be ser. See `protected function updateModel(event:UpdateCurrentModelEvent) : void` in cam.mxml

8.2.3.6 onSetToolbar

This event gives the tool a handle to the toolbar. It will be called at start-up and at every change of the active window. See public function `setToolBar(event:SetToolBarEvent):void` in `cam.mxml`. The availability status of buttons in the toolbar should be set, see public function `updateToolBar():void` in `cam.mxml`

8.2.4 **org.grapple_authoring_tool.authoring.ui**

This package contains the toolbar and widgets used by it. Some of these can be useful independently of the toolbar. For example `OpenFileDialog` is used in `cam.mxml` to allow the author to select which CRT (s)he wants to insert, see protected function `addCrt(evt:ContextMenuEvent):void` in `cam.mxml`.

The most important class for the GAT tools is `Toolbar.mxml`. It is used throughout `cam.mxml`, see the functions `updateSelection`, `setToolBar`, `getToolBar`, `updateToolBar`, `onAdded`, `load`, `save`, `newModel`, `menuItemClicked`, `updateModel`, `addCrt`, `confirmDeleteConcepts`.

The toolbar has the following public variables and functions:

Name	Use
static const MENU_FILE	Name (of type String) of menu item as passed in <code>onMenuItemClick</code>
static const MENU_FILE_NEW	
static const MENU_FILE_OPEN	
static const MENU_FILE_SAVE	
static const MENU_FILE_SAVE_AS	
static const MENU_FILE_PROPERTIES	
static const MENU_FILE_DELETE	
static const MENU_FILE_DEPLOY	
static const MENU_EDIT	
static const MENU_EDIT_SELECT	
static const MENU_EDIT_SELECT_ALL	
static const MENU_EDIT_CUT	
static const MENU_EDIT_COPY	
static const MENU_EDIT_PASTE	
static const MENU_EDIT_DELETE	
static const MENU_HELP_HELP	
static const MENU_VR	
static const MENU_SIMULATION	
static const MENU_HELP	
static const MENU_HELP_ABOUT	
var enabledMENU_FILE	Var that determines exaltedness (Boolean) of button in menu. Changes will only be active after a call to <code>setButtonEnabled</code> . Please call this after all variables have been set as repeated calls may cause a visible refresh delay.
var enabledMENU_FILE_NEW	
var enabledMENU_FILE_OPEN	
var enabledMENU_FILE_SAVE	

var enabledMENU_FILE_SAVE_AS	
var enabledMENU_FILE_PROPERTIES	
var enabledMENU_FILE_DELETE	
var enabledMENU_FILE_DEPLOY	
var enabledMENU_EDIT	
var enabledMENU_EDIT_SELECT	
var enabledMENU_EDIT_SELECT_ALL	
var enabledMENU_EDIT_CUT	
var enabledMENU_EDIT_COPY	
var enabledMENU_EDIT_PASTE	
var enabledMENU_EDIT_DELETE	
var enabledMENU_VR	
var enabledMENU_SIMULATION	
var enabledMENU_HELP	
var enabledMENU_HELP_HELP	
var enabledMENU_HELP_ABOUT	
var currentModel:XML=null;	The current model of the current active window, tools should set this variable when they receive an updateModelEvent
var container:Container;	Variable for use with popups. See protected function close() in basicPopUp.mxml
function reset():void	Rest the exaltedness of all buttons on the toolbar
function setButtonEnabled():void	Apply changes to the exaltedness of buttons Please call this after all variables have been set as repeated calls may cause a visible refresh delay.
function saveModel(model:XML, message:String="")	Save the model provided in <i>model</i> using the web services and display the message <i>message</i> upon success.
function addPopUp(popup:BasicPopup):void	Add the popup of type BasicPopup on top of the viewstack, temporarily disabling all underlying windows.

8.2.5 org.grapple_authoring_tool.authoring.utils

This package contains the following classes:

8.2.5.1 DrawingUtils

This class cannot be instantiated and contains the following public static functions that can be used throughout the GAT (importing of the class is still needed):

```
public static function dottedRect(graphics:flash.display.Graphics, fromX:Number, fromY:Number, toX:Number, toY:Number, gap:int=5, dot:int=5):void. Draws a rectangle with dotted or dashed line.
```

```
public static function dottedLine(graphics:flash.display.Graphics, fromX:Number, fromY:Number, toX:Number, toY:Number, gap:int=10, dash:int=5):void. Draws a dotted or dashed line, also used by dottedRect
```

`public static function stringToColour(colourString:String):uint.` Convert a string to uint. The function converts html style formats (e.g. #000000) to uint style formats (e.g. 0x000000)

8.2.5.2 KeyUtils

used by the menu for the keyboard shortcuts

8.2.6 **org.grapple_authoring_tool.authoring.CopyPaste**

Class for the copy-paste functionality. The variable model and items are available globally.

8.2.6.1 Concepts

For copy-pasting concepts, the model variable should contain a list of model(s) starting with the `<model>` tag. Often this will be a list of length 1 but items could come from different dm's.

The items array should contain a string array with the UUIDs of the items.

8.2.6.2 CRTS

For CRTs the items array is not used, the model XMLList is used in the same way as for the Concepts

8.2.7 **org.grapple_authoring_tool.authoring.GrappleVgraph**

An extension of the VisualGraph in the RaVis library that allows disabling traverse on double click and more importantly, selection of nodes and edges. Notable properties and functions are: `traverseOnDoubleClick`, `selectedNodes` (see `updateToolBar`), `selectedEdges` and `selectionMode` and `selectAll` (see `menuItemClicked`)

8.2.8 **org.grapple_authoring_tool.authoring.crtshapes**

These are the actual shapes a CRT can take, to add additional shapes, implement a shape along these lines and add to `shapes.xml`

8.2.9 **org.grapple_authoring_tool.authoring.socketshapes**

These are the actual shapes a socket can take, to add additional shapes, implement a shape along these lines and add to `shapes.xml`

8.2.10 **org.un**

Include of the ravis library