Reusing Components from Cultural Heritage Games – the RAGE Project Approach

K. Stefanov, B. Bontchev, P. Boytchev, A. Georgiev, A. Grigorov

Faculty of Mathematics and Informatics, Sofia University St Kl. Ohridski,
5, J. Bourchier Blv., Sofia, Bulgaria
{krassen, bbontchev, boytchev, alexander.arigorov,
atanas}@fmi.uni-sofia.bg

Abstract. Video games proved to be an attractive media for presenting cultural heritage issues. They can use virtual worlds to present cultural and historic information in amazing and highly appealing way. Each video game is implemented using set of valuable artifacts (components) reflecting technological, socioeconomic and historical issues. These software components are known as software assets, which are key output of the RAGE project. They can be used by game developers to enhance the pedagogical and educational value of their games. These software assets cover a broad spectrum of functionalities, including emotion detection, intelligent adaptation and social gamification. In order to facilitate integration and interoperability, all of these assets adhere to a common model, which describes their properties through a set of metadata. In this paper, we present the RAGE asset model and asset metadata model, highlighting key issues and challenges in constructing RAGE assets and using asset metadata model with flexible metadata editor, facilitating both adaptation and improvement of the asset metadata model.

Keywords: video games, cultural heritage, applied games, asset model, metadata model, metadata editor, gamification.

1 Introduction

Digital games are applied widely in various areas closely related to cultural heritage [1, 2]. They provide useful information and increasing appeal and engagement for all user ages, as well as fun and play [3]. Playing video games related to cultural heritage has proven benefits, providing cognitive, motivational, emotional and social benefits and leads to the development of related skills for the player [4]. These benefits, combined with embedded fun and motivation, make video games powerful tool for education and information about various cultural heritage issues.

In this paper, we show how video games related to cultural heritage can benefit from methods and tools developed in the frame of the RAGE (Realising an Applied Gaming Eco-system) project [5]. This project aims on the creation of an ecosystem for the development and exchange of software components (game assets), that could be incorporated in different games targeting different populations and application fields. Most of the available approaches and methodologies used for the development of applied games in cultural domain cannot be directly adopted, since they typically focus on design and development of domain-specific games, intended for specific game engines and platforms. In this context, RAGE project introduces additional stakeholders in game development processes - asset developers creating assets to be reused in different applied games and scenarios, either open source or paid assets.

A RAGE asset is a self-contained solution showing economic value potential, based on advanced technologies related to computer games, and intended to be reused in different game platforms. RAGE assets contain advanced game technology components (software), enriched and transformed to support applied games development. They go together with value adding services and attributes, like instructions, tutorials, examples and best practices, instructional design guidelines and methodologies, connectors to major game development platforms, data capacity, and content authoring tools/widgets for game content creation. Such artefacts should facilitate their usage.

RAGE assets are technologically based on an asset meta-model and framework for interoperability. RAGE assets are shared, exchanged and reused in the form of RAGE Asset packages. RAGE meta-model aims to define the structure of the RAGE Asset package, as well as the metadata format, used to describe different components of the RAGE Asset package. Ultimate goal of the document "Vocabulary and metadata categorization" is to propose the first version of this metadata format.

In the rest of this document, we enumerate the existing projects, major publications, standards, etc. used in order to make the relevant surveys and analysis, and to specify the RAGE project meta-model, as well as related metadata schema, and the framework and tools for their usage.

2 Related Research

We will present here model and taxonomy of existing games, in order to provide some clear classification of games, making it clear what kind of games will be supported in RAGE. We will also present taxonomy/ ontology describing the main game components/elements (based on the type of the game and game engine used; game engine components; graphics; video and animation; storyboarding; etc.).

A useful classification of applied games in relation to knowledge transfer from game to players was presented in [6]:

- Memory/Repetition/Retention (factual knowledge);
- Dexterity/Spread/Precision (sensorial knowledge);
- Applying Concepts/Rules (translating knowledge into new context);
- Decision-making (strategy and problem-solving);
- Social Interaction/values/cultures (understanding the social environment);
- The ability to learn/self-assessment (evaluation).

They define metadata schema for defining applied games, based on the IEEE LOM Profile, and centered on the following four main aspects of applied games:

- Gameplay - how the player interacts within a video game;
- Storytelling - theory (pedagogical and didactic specifications);
- Game design - technical specifications;
- Context - audience & application area.

Portugal defines *gameplay* as the combination of five components: Rules, Input methods, Space-related setup, Time-related setup, and Drama-related setup [7]. Gameplay types could be Game-based or Play-based. In [8] a set of 10 rules (called *combinable bricks*) are identified as key gameplay issues:

- Stated goals: Avoid, Match, Destroy;
- Means and constraints: Create, Manage, Move, Select, Shoot, Write, Random.

On the other hand, pedagogical scenarios and storytelling are two different but complementary forms of narration. They can be used to classify and describe cultural heritage games based on educational approaches and theories related to the science of learning used for the implementation of the learning scenario in the game.

There are different forms of the cultural game design, like world design (game story), system design (rules and logic), content design (characters, objects and context), game writing (dialogue and story), level design (levels, maps and challenges, user interface design (interaction, input and output information, feedback). A popular game design paradigm was proposed in [9] under the acronym MDA (Mechanics, Dynamics, and Aesthetics), where mechanics stand for game formal rules, their enforcement mechanisms, data representation and algorithms embedded within game components; dynamics describe the run-time behavior of the mechanics and aesthetics present emotional responses evoked in players by the dynamics like excitement, frustration or motivational intensity.

The Game Ontology Project (GOP) is a framework for describing, analyzing and studying games [10]. It is a hierarchy of concepts abstracted from an analysis of many specific games. The top level of the ontology consists of five elements: interface, rules, goals, entities, and entity manipulation. The ontology is represented as a hierarchy, so each element has parent and children. In addition, *part-of* relation is used to represent compound elements that are constructed out of other elements (parts). For example, Interface is parent for Input and Output, Input is parent for Input Method, Input Device, Manipulation Method, Locus of Manipulation, etc.

A theory defines nine possible element categories that are found throughout the universe of games [11]. The categories are explained below, proceeding from simpler elements to the more complex.

1. Components: The resources for play; what is being moved or modified - physically, virtually, in transactions - in the game, between players and the system. Tokens, tiles, balls, characters, points, vehicles are common examples of game components.
2. Environment: The space for play – boards, grids, mazes, levels, worlds.
3. Ruleset: The procedures with which the game system constrains and moderates play, with goal hierarchy as an especially important subset.

4. Game mechanics: What actions the players take as means to attain goals when playing. Placing, shooting, maneuvering are examples of what players are put to perform in many games.
5. Theme: The subject matter of the game, which functions as a metaphor for the system and the ruleset.
6. Information: What the players need to know and what the game system stores and presents in game states: Points, clues, time limits, etc.
7. Interface: In case there are no direct, physical means for the player to access game elements, interface provides a tool to do that.
8. Players: Those who play, in various formations and with various motivations, by performing game mechanics in order to attain goals.
9. Contexts: Where, when, and why the gaming encounter takes place.

By minimum, a game has to have Components, Environment, and at least one Game Mechanic. When the relationships of these three elements are defined and implemented, it means that a Ruleset emerges, as does Information. Then we need Players, and any gaming encounter brings about various Contexts, that may vary from one encounter to the next one.

3 The RAGE Metadata Model

A RAGE asset as a self-contained software component related to computer games, intended to be reused in different game platforms. The RAGE asset is designed to contain advanced game technology (software), as well as value-adding services and attributes that facilitate their use, e.g. instructions, tutorials, examples and best practices, instructional design guidelines, connectors to major game development platforms, test plans, test scripts, design documents, data capacity, and content authoring tools/widgets for game content creation.

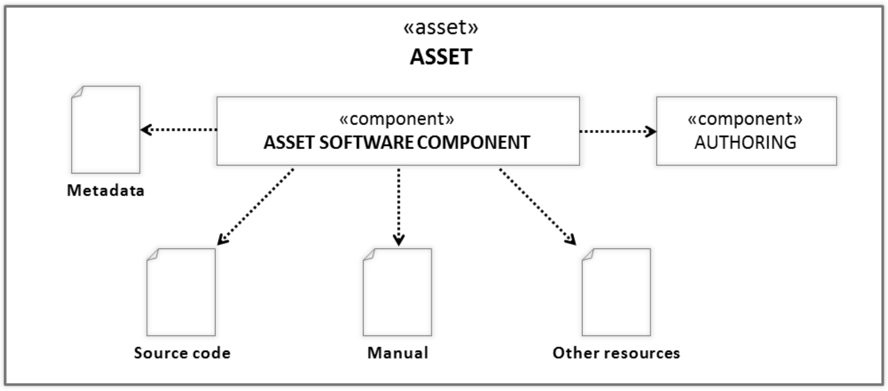


Fig. 1. Conceptual layout of a RAGE asset.

Fig. 1 presents the general layout of a RAGE asset. Its software architecture is component-based and has been described and validated in [12], [13], and [14]. It addresses both the internal workings of an asset and the level of interaction of assets with the outside world, including the mutual communications between assets. The RAGE architecture avoids dependencies on external software frameworks and minimises code that may hinder integration with game engines. It relies on a limited set of standard software patterns and well-established coding practices.

The RAGE ecosystem is more complex than a typical Learning Object exchange ecosystem, with different types of products (generic assets, asset instances and playable games) co-existing and using the same metadata standards. For this reason, we propose a layered approach, in which assets are marked with specific metadata that describe the asset from technical and (if appropriate) cultural heritage perspectives. These metadata should cover the requirements for discovery and configuration in the asset repository and comply with the additional requirements derived from the general asset architecture. Then, metadata from the assets would be incorporated in the metadata of the asset instances, and on the metadata of the games created with that asset instance.

After careful analysis of the needs of the stakeholders [15], we decided to create our metadata around the following main fields:

- Asset main goal (can be expressed as properties and rules related to game contexts, game and learning mechanics, learning functions, etc.);
- Asset high-level solution description (including properties related to additional artefacts like requirements, models, code, tests, documents, etc.);
- Game context definition (in what situations and game scenarios the asset can be linked and applied – domain, game and learning contexts, performance indicators values, etc.);
- Asset as game element [10, 11] - defines the asset as a game component;
- Asset as learning element: matching game mechanics to learning mechanics, following GALA model and its simplification aligned to Bloom's classification [16, 17];
- Asset as educational game pattern (following [18]);
- Learning functions of the Asset (following [19]);
- Asset usage (instructions, documentations, parameters, initializations, data exchange, etc.);
- Asset relations (relations with other assets like aggregation, dependency, parent, association, etc.).

4 Designing a Shared Metadata Set

In order to support a wide set of services through the software repository and other related tools and, in parallel, to be close to the specified domain of reusable gaming components (RAGE software assets), the RAGE metadata model is focussed on the following main aspects:

- *Technical* – how the RAGE asset might be used by game developers. We follow the RAGE asset model which describes assets as software components according to software engineering standards.
- *Contextual classification* – here we focus primarily on the pedagogical, educational and game characteristics, whilst leaving space for further characteristics.
- *Usage* – not restricted to how to install and configure the software, but also providing additional artefacts such as training materials, tutorials, educational goals, etc.

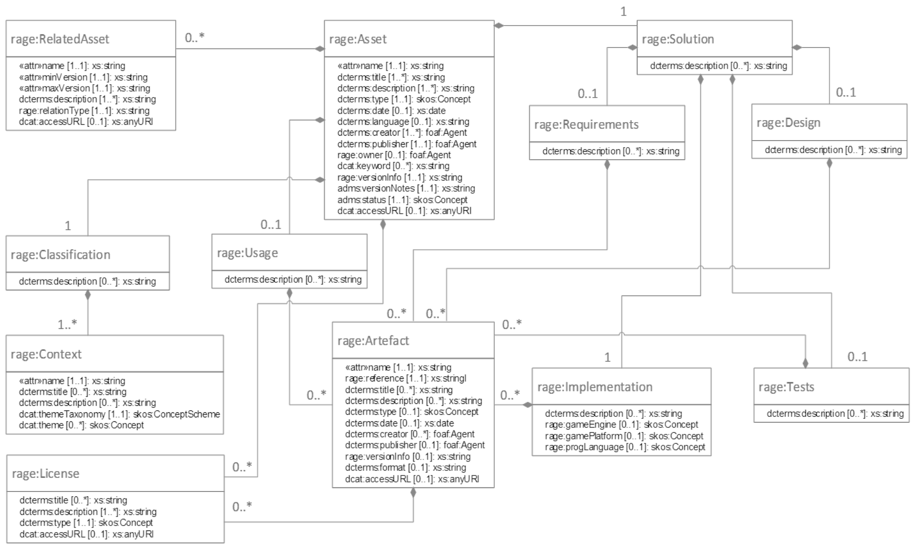

rage:RelatedAsset
«attr»name [1..1]: xs:string
«attr»minVersion [1..1]: xs:string
«attr»maxVersion [1..1]: xs:string
dcterms:description [1..*]: xs:string
rage:relationType [1..1]: xs:string
dcat:accessURL [0..1]: xs:anyURI
0..*
rage:Asset
«attr»name [1..1]: xs:string
dcterms:title [1..*]: xs:string
dcterms:description [1..*]: xs:string
dcterms:type [1..1]: skos:Concept
dcterms:date [0..1]: xs:date
dcterms:language [0..1]: xs:string
dcterms:creator [1..*]: foaf:Agent
dcterms:publisher [1..1]: foaf:Agent
rage:owner [0..1]: foaf:Agent
dcat:keyword [0..*]: xs:string
rage:versionInfo [1..1]: xs:string
adms:versionNotes [1..1]: xs:string
adms:status [1..1]: skos:Concept
dcat:accessURL [0..1]: xs:anyURI
1
rage:Solution
dcterms:description [0..*]: xs:string
0..1
rage:Requirements
dcterms:description [0..*]: xs:string
0..1
rage:Design
dcterms:description [0..*]: xs:string
1
rage:Classification
dcterms:description [0..*]: xs:string
0..1
rage:Usage
dcterms:description [0..*]: xs:string
1..*
rage:Context
«attr»name [1..1]: xs:string
dcterms:title [0..*]: xs:string
dcterms:description [0..*]: xs:string
dcat:themeTaxonomy [1..1]: skos:ConceptScheme
dcat:theme [0..*]: skos:Concept
0..*
0..*
0..*
rage:Artefact
«attr»name [1..1]: xs:string
rage:reference [1..1]: xs:stringl
dcterms:title [0..*]: xs:string
dcterms:description [0..*]: xs:string
dcterms:type [0..1]: skos:Concept
dcterms:date [0..1]: xs:date
dcterms:creator [0..*]: foaf:Agent
dcterms:publisher [0..1]: foaf:Agent
rage:versionInfo [0..1]: xs:string
dcterms:format [0..1]: xs:string
dcat:accessURL [0..1]: xs:anyURI
0..*
0..*
1
rage:Implementation
dcterms:description [0..*]: xs:string
rage:gameEngine [0..1]: skos:Concept
rage:gamePlatform [0..1]: skos:Concept
rage:progLanguage [0..1]: skos:Concept
0..1
rage:Tests
dcterms:description [0..*]: xs:string
rage:License
dcterms:title [0..*]: xs:string
dcterms:description [1..*]: xs:string
dcterms:type [1..1]: skos:Concept
dcat:accessURL [0..1]: xs:anyURI
0..*
0..*

Fig. 2. The RAGE metadata schema

- *Intellectual Property Rights* – to enable various business models proposed by RAGE project to be implemented.

Whilst designing the model, we broadly adhered to the following general metadata design principles:

- *Reusability* – reusing where possible existing metadata models, standards and available taxonomies and ontologies. We reuse parts of the RAS [20] and ADMS [21] metadata fields as well as parts of the LOM taxonomy [6].
- *Flexibility* – easy facilitation of extension of the metadata description of an asset with additional features and characteristics.
- *Simplicity* – we define most of the fields as not mandatory in order to easy the efforts of asset developers.

The RAGE asset metadata model reuses and extends the specifications of RAS [20] and ADMS [21]. We have chosen this approach to use a core subset of RAS and extend it with elements from ADMS, IEEE LOM and metadata related to the applied games domain.

The RAGE Metadata Model defines the format of asset metadata as an XML schema, which will be implemented in all tools that require the processing of these metadata, e.g. a package metadata editor, the asset repository, asset installation widgets, etc. The XML [22] schema represented as a UML [23] class diagram is displayed in Fig. 2. The figure can be interpreted as follows:

- A UML class represents a complex XML element. For example, «Asset» is a complex XML element that has XML attributes and/or child elements.
- Within a UML class definition, the prefix «attr» denotes that the corresponding field is an XML attribute (and not a child element).
- Within a UML class definition, a field without «attr» prefix denotes a child element nested inside the element represented by the UML class definition. The field represents either a new definition of a simple element, or a reference to an element (either simple or complex) defined in a referenced schema (e.g., «dcterms»).
- Composition connection denotes a parent-child relationship between two complex elements defined in the RAGE Metadata Schema.

Table 1 provides a textual description of the XML elements of the RAGE metadata schema.

Table 1. Description of RAGE metadata schema elements.

Asset	A self-contained solution that demonstrates economic value potential, based on advanced technologies related to computer games, and intended to be reused or repurposed in a variety of game platforms and scenarios.
Classification	Includes a set of descriptors for classifying the asset as well as a description of the context(s) for which the asset is relevant.
Solution	Describes the artefacts of the asset.
Usage	Contains information for asset installation, customization, and use.
Related Assets	Describes the asset's relationship to other assets.
Artefact	Any physical element of an asset corresponding to a file on a file system. Artefacts can include also version and license information.
Requirements	Contains artefacts that specify the asset requirements, such as models, use cases, or diagrams.
Design	Contains artefacts that specify the asset design such as diagrams, models, interface specifications, etc.
Implementation	Has a collection of artefacts that identify the binary and other files that provide the implementation.
Tests	Contains artefacts (models, diagrams, artefacts, and so on) that are intended to describe the testing of the asset such as testing procedures, concerns and test units.
License	Contains conditions or restrictions that apply to the use of an

	asset or artefact, e.g. whether it is in the public domain, or that some restrictions apply such as attribution being required, or that it can only be used for non-commercial purposes, etc.
Context	Defines a conceptual frame, which helps explain the meaning of other elements in the asset.
Custom Metadata	An element that serves as an extension point for adding custom metadata as name-value pairs.
Agent	Describes a person or organization that is a contributor (creator, publisher, and owner) of an asset or artefact.
Concept Scheme	A vocabulary, thesaurus or taxonomy used for organizing concepts.
Concept	Represents a particular concept within a vocabulary, thesaurus or taxonomy.

5 Asset Repository Infrastructure

The RAGE asset software repository is at the core of the asset development infrastructure. It is used to store and manage access to:

- Reusable game assets
- Artefacts – resources within game assets
- Metadata for game assets and for artefacts
- Relationships between assets – dependencies, related assets, etc.

The Asset software repository leverages the discovery, development reuse and repurpose of game assets and artefacts. It helps both game asset developers and consumers in all the activities relating to the game asset lifecycle. The main functions of the RAGE Asset software repository are as follows:

- Search for assets and artefacts in the repository
- Find related assets/artefacts
- Browse assets/artefacts
- Create, update, publish, delete assets/artefacts
- Download assets/artefacts
- Versioning support
- Source code import from GitHub - using the GitHub API [24]
- Integration with IDEs
- Harvesting of repositories for game assets and metadata using the Open Archives Initiative - Protocol for Metadata Harvesting [25]
- Review and rate assets/artefacts
- Subscription to assets

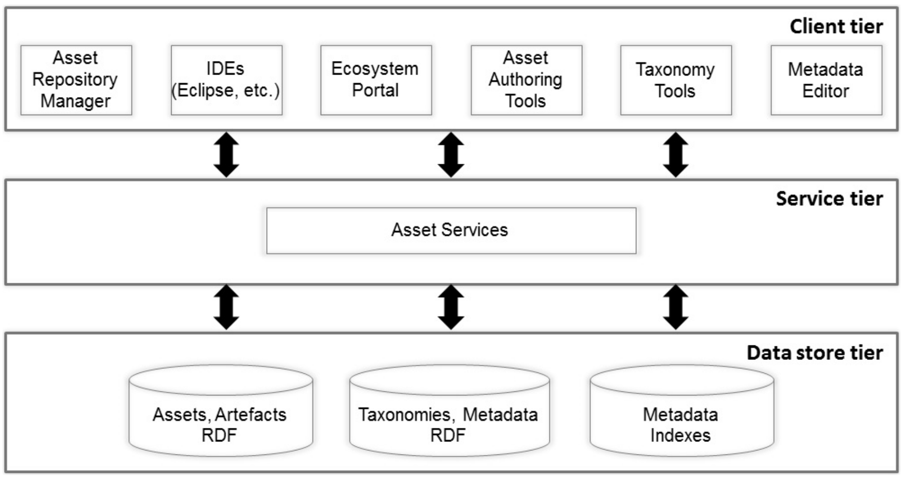


Fig. 3. The Asset Repository Architecture.

The asset repository infrastructure comprises three tiers (Fig. 3): client, service and data store tiers. Based on the functionality exposed by the services, they can be grouped as follows:

- *Asset Access Services* defining an open interface for accessing assets within the RAGE Asset Repository. The access operations include retrieving asset packages and metadata, search and browse for assets using keywords and metadata fields. The search interface provides both full-text search and semantic search. Full-text search enables performing of natural language queries using keywords and phrases occurring in any of indexed asset's metadata elements. The semantic search is using queries on asset metadata and taxonomies.
- *Asset Management Services* defining an open interface for administering assets, including creating, modifying, and deleting. These services interact with the underling services providing an abstract level of the operations and thus hiding the complexity about the internal formats, protocols and procedures for storing an asset in the Asset Repository.
- *Taxonomy Services* defining an open interface for managing classification taxonomies and controlled vocabularies used in RAGE Asset Metadata Model to classify and describe an asset in educational and gaming contexts.
- *Authentication and Authorization Services* – provide an extensible and configurable access best suiting organisational needs. These services use OpenId [26] authentication provider and mapping to one or more preconfigured roles, respectively.

An Asset is packaged into an Asset Package, which includes the asset code (the asset software component) combined with additional resources such as meta-data, background information, manual, installation test suit, configuration/authoring tools, tutorials, scientific data or tool with auxiliary functionalities. Packaging is the process of combining all files of an asset into an easy to find and easy to download entity. A game developer is able to (1) search the Repository for an asset, (2) download it as a Package, (3) extract the asset code and other artefacts from the Package, and (4) integrate/install the asset software.

An XML Schema (described in DefaultProfile.xsd) in accordance with the RAGE Metadata model has been developed. It defines a standardized machine-readable syntax for creating Package Metadata. Each Package Metadata should be validated against the Package Schema to verify compliance with RAGE distribution standards.

The Package Interchange File (PIF) is a unit of distribution a RAGE Asset Package represented by a ZIP archive containing manifest, metadata and artefacts (see Fig. 4).

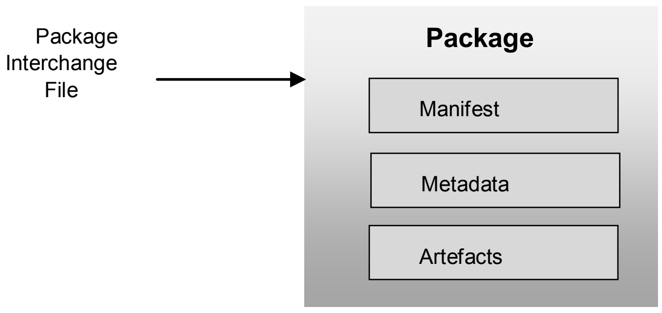


Fig. 4. The Asset Package.

6 Front-end Tools to Access the Software Asset Repository

6.1 RAGE Taxonomy tools

Taxonomies used in RAGE's asset metadata schema represent hierarchies of concepts and controlled vocabularies. They provide prefixed sets of values for some metadata elements. The front-end taxonomy tools in RAGE are actually a set of three tools – *Taxonomy Viewer, Taxonomy Selector* and *Taxonomy Editor*. They are generic tools written in JavaScript that share a common core and can be embedded into a web page.

The design principles of the taxonomy tools are: (1) minimalistic user interface and set of features; (2) consistent visual layout; (3) multiplatform support through HTML5 and JavaScript; and (4) multilingual interface and taxonomy content. The taxonomy tools are part of the client tier and they access the repository through the asset services of the services tier.

The RAGE Taxonomy Viewer is used to browse taxonomies. It provides several layouts and interface languages. The RAGE Taxonomy Selector is used to pick concepts from taxonomies. This tool supports the functioning of other front-end tools, such as the metadata editor, asset configuration tools and various authoring tools. Finally, the RAGE Taxonomy Editor, is used to edit taxonomies. It supports structural changes to a taxonomy (i.e. adding, deleting and reallocating concept nodes around the hierarchy) and content changes (i.e. changing concept names and concept translation in several languages).

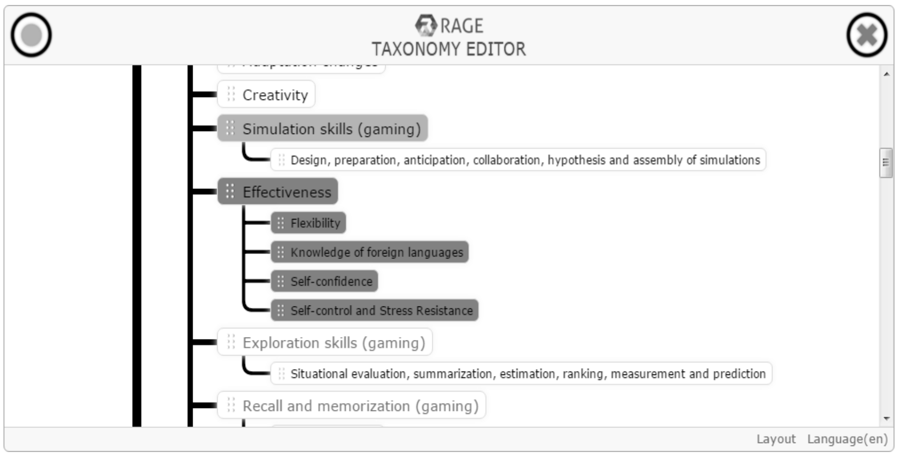
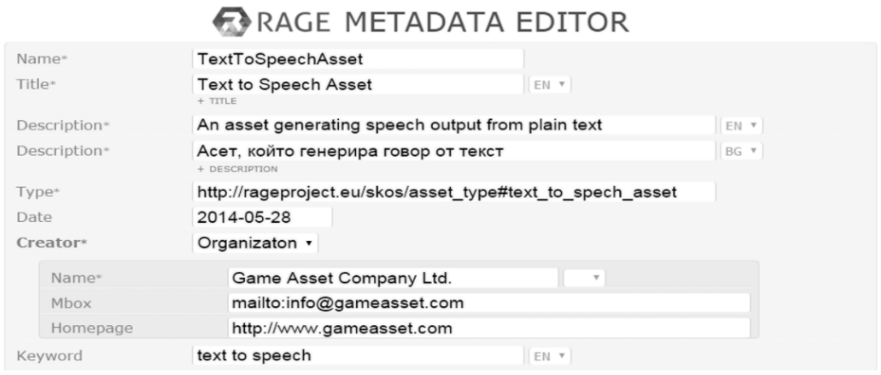

RAGE
TAXONOMY EDITOR
Creativity
Simulation skills (gaming)
Design, preparation, anticipation, collaboration, hypothesis and assembly of simulations
Effectiveness
Flexibility
Knowledge of foreign languages
Self-confidence
Self-control and Stress Resistance
Exploration skills (gaming)
Situational evaluation, summarization, estimation, ranking, measurement and prediction
Recall and memorization (gaming)
Layout
Language(en)

Fig. 5. A snapshot of the RAGE Taxonomy Editor.

6.2 The RAGE Metadata Editor

Another front-end tool is the RAGE *Metadata Editor*. It provides functionality to edit the metadata associated with an asset or another RAGE metadata entity. The editor hides the internal metadata complexity and constructs a flexible dynamic interface as shown in fig. 6.

When the editor loads a metadata file, it extracts the schema definition of the metadata and builds the user interface. The hierarchy of metadata is represented as nested blocks which nesting goes as deep as it is defined by the schema. When the interface is constructed, the editor extracts the actual metadata and populates them in the interface. This may recursively trigger a partial reconstruction of the interface for nodes of cardinality higher than 1. After the metadata are edited by the user, the editor generates an XML representation of the metadata and sends it back to the server.



Name*	TextToSpeechAsset	
Title*	Text to Speech Asset	EN
Description*	An asset generating speech output from plain text	EN
Description*	Асет, който генерира говор от текст	BG
Type*	http://rageproject.eu/skos/asset_type#text_to_spech_asset	
Date	2014-05-28	
Creator*	Organizaton	
Name*	Game Asset Company Ltd.	
Mbox	mailto:info@gameasset.com	
Homepage	http://www.gameasset.com	
Keyword	text to speech	EN

Fig. 6. The RAGE Metadata Editor.

7 Conclusions

In this paper, we present a unique approach for designing software architecture supporting the lifecycle of reusable software components – game assets. We have built on the best practices described in the research literature, including RAS based asset repositories, metadata based educational repositories, and game asset stores. The software architecture plays a pivotal role within the RAGE Ecosystem, developed for the RAGE project.

The repository as the content core system of the RAGE Ecosystem allows for flexible design and development of RAGE game assets and future search, packaging and exchange. The current architecture guarantees both scalability and durability and the approach provides a high level of flexibility across different taxonomies and standards.

Future work is planned on improving the architecture by providing support for Quality Assurance, asset development workflows, harvesting of assets from external systems and stores, social functions and for specific targeted support for the gaming community.

Acknowledgements. This work has been partially funded by the EC H2020 project RAGE (Realising an Applied Gaming Eco-System); http://www.rageproject.eu/; Grant agreement No 644187.

References

1. Sotirova, K., Peneva, J., Ivanov, S., Doneva, R., Dobreva, M.: Digitization of Cultural Heritage–Standards, Institutions, Initiatives. Chapter 1, Access to Digital Cultural Heritage: Innovative Applications of Automated Metadata Generation, ISBN: 978-954-423-722-6, Plovdiv, Bulgaria (2012)
2. Zyda, M.: From Visual Simulation to Virtual Reality to Games. IEEE Computer, Sept., Vol. 38 (9), pp.25-32 (2005)
3. Anderson, E. F., McLoughlin, L., Liarokapis, F., Peters, C., Petridis, P., & Freitas, S.: Serious games in cultural heritage, Proc. of 10th Int. Symp. on Virtual Reality, Archaeology and Cultural Heritage VAST (2009)
4. Bontchev, B.: Serious Games for and as Cultural Heritage, in Proc. of the Fifth Int. Conf. on Digital Presentation and Preservation of Cultural and Scientific Heritage, UNESCO, Issue No. 5, Veliko Tarnovo, Bulgaria, Sept. 28–30, pp.43-58 (2015)
5. RAGE: Project Web site, http://www.rageproject.eu, last visited 30/06/2016
6. El Borji, Y., Khaldi, M.: An IEEE LOM Application Profile to Describe Serious Games, International Journal of Computer Applications, Volume 86 – No 13, January, (2014)
7. Portugal, J-N.: Le Rapprochement du Jeu et de l'Apprentissage. Serious Games Summit Europe, Lyon, France (2006)
8. Djaouti, D., J. Alvarez, J. Jessel, G. Methel, P. Molinier: A Gameplay Definition through Videogame Classification, International Journal of Computer Games Technology, Vol 2008, Article ID 470350, http://dx.doi.org/10.1155/2008/470350 (2008)
9. Hunicke, R., LeBlanc, M., Zubek, R.: MDA: A formal approach to game design and game research. Proc. of the AAAI Workshop on Challenges in Game AI, Vol. 4, July, (2004)

10. Zagal, J., Mateas, M., Fernandez-Vara, C., Hochhalter, B. and Lichti, N.: Towards an Ontological Language for Game Analysis, Proc. of the Digital Interactive Games Research Association Conference (DiGRA 2005), Vancouver B.C., June, (2005)
11. Järvinen, A.: Introducing Applied Ludology: Hands-on Methods for Game Studies. in Situated Play, Proc. of DiGRA 2007 Conference, ACM, pp. 134-144, (2007)
12. Van der Vegt, W., Nyamsuren, E., Westera, W.: RAGE Reusable Game Software Components and Their Integration into Serious Game Engines. In: Georgia M. Kapitsaki and Eduardo Santana de Almeida (Eds.), Bridging with Social-Awareness, Proc. of 15th Int. Conf. ICSR 2016, Limassol, Cyprus, June 5-7, 2016, LNCS, Vol. 9679, pp. 165-180 (2016)
13. Van der Vegt, W., Westera, W., Nyamsuren, E., Georgiev, A., Martínez Ortiz, I.: RAGE Architecture for Reusable Serious Gaming Technology Components, Int. Journal of Computer Games Technology, doi:10.1155/2016/5680526 (2016)
14. Westera, W., Van der Vegt, W., Bahreini, K., Dascalu, M., Van Lankveld, G.: Software Components for Serious Game Development. Proc. of 10th European Conf. of Game-Based Learning, October 6-7, 2016, Paisley, Scotland (2016)
15. Saveski, G. L., Westera, W., Yuan, L., Hollins, P., Fernández Manjón, B., Moreno Ger, P., Stefanov, K.: What serious game studios want from ICT research: identifying developers' needs. In: A. De Gloria and R. Veltkamp (Eds.), Proc. of the GALA 2015 Conf., December 7-8, Rome, Italy, LNCS 9599, pp. 1–10 (2015)
16. De Freitas, S., Oliver, M.: A four dimensional framework for the evaluation and assessment of educational games, Computer Assisted Learning, (2005)
17. Bloom Games Taxonomy: Free Taxonomy Alignment for Gaming, Allen Interactions, http://outreach.alleninteractions.com/poster-taxonomy-alignment-for-gaming-tag/ (2014)
18. Björk, S., Holopainen, J.: Patterns in Game Design, Charles River Media (2004)
19. Kiili, K.: Call for learning-game design patterns. Edvardsen, F. & Kulle, H.(eds.). Educational games: design, learning and applications, Nova Publishers (2010)
20. OMG: Reusable Asset Specification (RAS), Version 2.2, Release Date: November 2005, http://www.omg.org/spec/RAS/2.2/, (2005)
21. ADMS: Asset Description Metadata Schema (ADMS), W3C proposal standard, (2013)
22. XML: XML 1.0 Specification, World Wide Web Consortium, (2010)
23. UML: Unified Modeling Language (UML), revision 2, ISO/IEC 19505-1:2012, (2012)
24. GitHub API: GitHub Developer Guide (2016) https://developer.github.com/v3/
25. Lagoze, C., Van de Sompel, H.: The Open Archives Initiative Protocol for Metadata Harvesting, https://www.openarchives.org/OAI/openarchivesprotocol.html (2015)
26. OpenID: OpenID Connect Specification, retrieved from http://openid.net/developers/specs/ (2014)