The RAGE Advanced Game Technologies Repository for Supporting Applied Game Development

A. Georgiev[1], A. Grigorov[1,6], B. Bontchev[1], P. Boytchev[1], K. Stefanov[1], W. Westera[2], R. Prada[3], Paul Hollins[4], Pablo Moreno[5]

[1] Sofia University "St. Kliment Ohridski", Faculty of Mathematics and Informatics, Bulgaria
{atanas,alexander.grigorov,bontchev,boytchev,stefanov}@fmi.uni-sofia.bg
[2] Open University of the Netherlands
Wim.Westera@ou.nl
[3] University of Lisbon, Portugal
rui.prada@tecnico.ulisboa.pt
[4] The University of Bolton, UK
pah1@bolton.ac.uk
[5] Universidad Complutense de Madrid, Spain
pablom@fdi.ucm.es
[6] Institute of Mathematics and Informatics, Bulgarian Academy of Sciences, Bulgaria
grigorov@math.bas.bg

Abstract. This paper describes the structural architecture of the RAGE repository, which is a unique and dedicated infrastructure that provides access to a wide variety of advanced technologies (RAGE software assets) for applied game development. These software assets are reusable across a wide diversity of game engines, game platforms and programming languages. The RAGE repository allows applied game developers and studios to search for software assets for inclusion in applied games. The repository is designed as an asset life-cycle management system for defining, publishing, updating, searching and packaging for distribution of these assets. The RAGE repository provides storage space for assets and their artefacts. It will be embedded in a social platform for networking among asset developers and other users. A dedicated Asset Repository Manager provides the main functionality of the repository and its integration with other systems. Tools supporting the Asset Manager are presented and discussed. When the RAGE repository is in full operation, applied game developers will be able to easily enhance the quality of their games by including advanced game technology assets.

Keywords: software assets, serious games, asset repository, asset development, taxonomy tools, metadata editor, applied games, reuse.

1 Introduction

Applied gaming is highlighted as one of the main priorities in Horizon2020, the Research and Innovation Programme of the European Commission. Policy makers of the

European Commission envision a flourishing applied games industry that helps to address a variety of societal challenges in education, health, social cohesion and citizenship, and equally one that stimulates the creation of jobs in the creative industry sector.

Although applied or serious games have been successfully employed in education and training settings across a wide and varied range of application domains, seizing the full potential of applied games has been challenging. In contrast, the leisure games industry is an established industry dominated by large international hardware vendors (e.g. Sony, Microsoft and Nintendo) and large publishers and retailers. Conversely, the applied game industry is fragmented across a large number of small independent businesses with limited interconnectedness and knowledge exchange [1, 2].

The RAGE project [3] aims to stimulate the applied game industry by making available a set of advanced reusable game technology components (software assets) that game studios can easily integrate in their game development projects. Applied game studios would benefit from using state-of-the-art technologies, while incorporating complex pedagogic technical functionality would become easier and quicker, and the cost of development would be reduced. The software assets cover a variety of functionalities including game analytics, emotion recognition, assessment, personalised learning, game balancing and player-centric adaptation, procedural animation, language technologies, interactive storytelling, and social gamification.

While the main research goal of the RAGE project is to support the applied game industry by making available a large set of reusable, advanced software components (applied gaming assets), this paper focuses on the design of the repository infrastructure that supports the processes of development, reuse and sharing of applied gaming assets. This paper presents the asset repository architecture and the associated asset development methodology. We first present the related work efforts, then discuss our approach (research method), describe the software asset concept, provide details of the design and implementation of the back-end repository system architecture and corresponding front-end tools, and we conclude with a brief description of first experiments with the infrastructure, analysis and identification of further development and research efforts.

2 Related work

Asset-based software development relies on reusing well documented and cohesive software artefacts and, therefore, it is inconceivable without a platform for storing and accessing assets. An asset repository as a software tool is defined by Ackerman and colleagues [4] for storing and retrieving reusable assets and managing asset access control for asset producers and consumers, according to the phases of the asset life cycle. They introduce the IBM Rational Asset Manager (RAS) repository, which handles tasks and activities of software asset producer, consumer and subscriber roles, while offering reduced production costs and improved software quality. In order to facilitate cross-project reuse of assets, the Rational Asset Manager model provides monitoring of asset categorization and usage together with multi-platform compliance management.

Another example for a RAS-based asset repository is the Atego Asset Library [5], which is a scalable Web-based repository for reusable software engineering artefacts. It is based on OMG RAS and integrates Unified Modelling Language (UML) and Systems Modelling Language (SysML) in order to facilitate asset reuse at design time.

Currently, the tool is supported as PTC Integrity Asset Library[1] and, besides the publishing, finding and reuse of assets, provides services as interest registry and notification, automatic file interrogation, traceable links and reuse metric dashboard.

Extensions of the OMG RAS have been proposed for designing open source Web-based asset repositories providing advanced classification, search and utilization of reusable software assets of various types. The OpenCom asset repository was created as a supporting tool of Shanghai Component Library [6] based on an extension of OMG RAS profile aiming at collaborative creation of knowledge by web users. The Lavoi free source asset repository [7] was developed based on an extension of the component profile of OMG RAS broadening the categories about classification, solution, usage and related assets.

Within the computer games domain, the *asset* concept is often reserved for media files to be incorporated in a game. For example, the Intel® XDK HTML5 Cross-platform Development Tool [8] offers an asset manager for game development in conjunction with several game platforms. Here assets are often considered audio-visual game objects to be included in a project. In RAGE the focus is on software assets, reusable components adding specific (pedagogic) functionality for applied game development.

A similar attempt related to using a digital repository of metadata resources for education, combined with a portal for the respective community of practices build around the repository, is described in [9]. Other approaches to endowing digital libraries with adaptability capabilities in order to scaffold and enhance end user experience are presented in [10]. Similar attempts inside GALA Network of Excellence are the SoA framework for SGs [25] and the repository for exchange of game resources [26].

3 RAGE Software Assets

A RAGE asset as a self-contained software component related to computer games, intended to be reused and or repurposed across different game platforms. Its formal definition is compliant with the asset definition of the W3C ADMS Working Group [11], which refers to abstract entities that reflect some “intellectual content independent of their physical embodiments”. In principle, not all assets are required to include software, however this paper focusses on software assets.

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.

[1] http://www.ptc.com/model-based-systems-engineering/integrity-modeler/asset-library

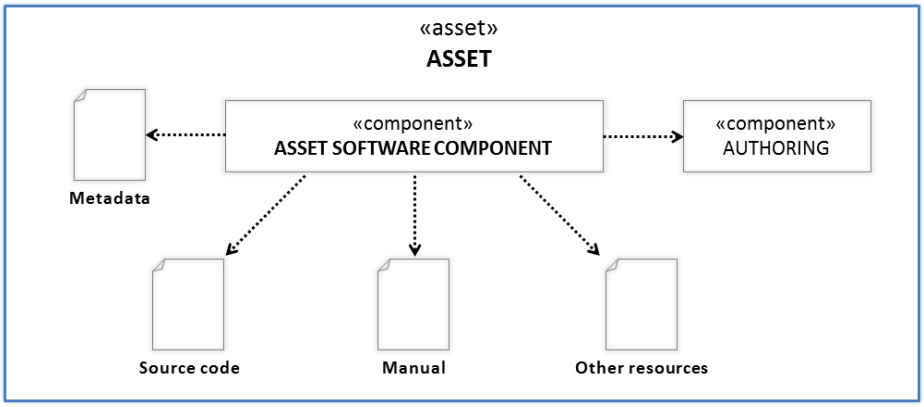

Figure 1. Conceptual layout of a RAGE Asset

Figure 1 presents the general layout of a RAGE asset. Its software architecture is component-based and has been described and validated in [12]. 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. Each RAGE asset contains metadata, which describe its content and functionality. RAGE metadata model in the domain of applied gaming was designed for defining the asset's metadata and for enabling the proper implementation of the RAGE Asset repository system architecture [13].

4 Our approach

The research methodology for this study is based on the Rapid Application Development model [14]. We performed an extensive needs assessment study [15], including asset developers, educators and game producers. We have identified the services to be supported through the repository and other related tools and, in parallel, designed the RAGE metadata model to fit the specified domain of reusable gaming components (RAGE software assets). It was clear that we could not reuse any existing solution, but needed to design and implement our own software repository, targeting the identified needs and characteristics of the applied game domain.

In the next stage we provided the initial design of the RAGE asset as a software component, and the architecture of the RAGE software repository, aimed at supporting the development, storage, sharing and reuse of assets. In the next stage we provided details on the technical implementation of the software repository. We performed several interactions between these two stages until we reached a stable and more or less complete solution. In the last stage we analysed the first use case scenarios of the repository through several client tools, arranged first evaluations of the repository, and collected ideas for its improvement in the next cycle.

We will present the results of each stage in the next sections.

5 The Asset repository system architecture

Metadata is a key part of the information infrastructure necessary to help create order and provide a solid foundation for providing various information services such as descriptions, classifications, organizations, store, search, creation, modification and aggregation of information [16]. Rather than merely a software archive, the asset repository is viewed as a system for managing the lifecycle of an asset. In the repository the asset's artefacts are collected and conceptually tied together by defining the metadata. In addition, the repository allows for publication, updating, packaging for distribution and quality assurance, while accommodating different end-user tools.

The RAGE asset software repository is at the core of the asset development infrastructure. It is used to store and manage access to: (1) reusable game assets, (2) artefacts (resources within game assets), (3) metadata for game assets and artefacts, and (4) relationships between assets – dependencies, related assets, etc.

The Asset software repository leverages the discovery, development reuse and repurpose of game assets and artefacts. It will help 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:

- Searching, finding and browsing assets/artefacts
- Creating, updating, publishing, deleting and downloading assets/artefacts
- Versioning support, source code import from GitHub and integration with IDEs
- Harvesting of external repositories for game assets and metadata using the Open Archives Initiative - Protocol for Metadata Harvesting (OAI-PMH)
- Reviewing and rating assets/artefacts

In order to implement these functions, we designed the asset repository infrastructure in three tiers (Figure 2): client, service and data store tiers.

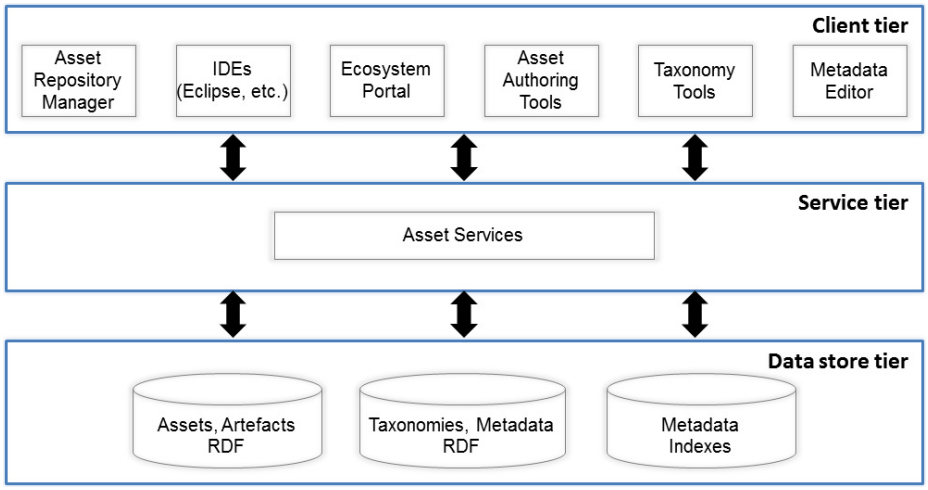

Figure 2. Asset Repository Architecture

6 Implementation of the asset repository system architecture

The main result from the second stage – Acting, is the implementation of the Asset repository. Fedora [17] is used for storing assets, metadata and artefacts; Sesame [18] for managing RDF data and supporting classification and entities; and Solr [19] for indexing and searching the repository. The data store tier consists of these three components and is used to store game assets, artefacts, metadata, taxonomies and indexes:

- **Fedora** stores the game assets, artefacts and metadata using RDF as primary data format. When the repository is updated by creating, modifying or deleting resources, it generates specific events so that the Fedora indexer copies RDF from the repository to an external triple store to keep it synchronized with the repository. Fedora is flexible, well established and it ensures scalability and durability (the complete repository can be rebuilt at any time).
- **Sesame** is an architecture for the efficient storage and expressive querying of large quantities of metadata in RDF and RDF Schema. This includes creating, parsing, storing, inferencing and querying over such data. Sesame RDF triple store contains metadata from Fedora and classification taxonomies/vocabularies.
- **Solr** is an open source platform optimized for searching. Its major features are full-text search, sophisticated faceted search, almost real-time indexing, dynamic clustering of data, etc. It is used for creating full text indexes on the RAGE metadata fields, as well as for realizing full text search and faceted search.

The service tier is used for access and preservation of the assets and artefacts. For the implementation of this tier, we developed the following services that provide access to the underlying data store tier:

- **Fedora Services**. Fedora provides a general RESTful HTTP API for accessing repository resources through HTTP methods. It supports OAI-PMH [20] requests on content and metadata in the repository.
- **Sesame Services**. Sesame offers a RESTful HTTP interface supporting the SPARQL Protocol for RDF. It is a superset of the SPARQL and supports communication for Update operations and the Graph Store HTTP Protocol [21].
- **Solr Services**. Apache Solr exposes Lucene's Java API as REST-like API's which can be called over HTTP. The RESTful endpoints allow CRUD style operations to be performed on the repository resources.

In addition, for the service tier to provide access to the client tier, we developed **Asset Services** for composition and execution of workflows over RAGE Game Assets.

The client tier includes web-based applications, plug-ins for integrated development environments (IDEs), and software components from the RAGE ecosystem that uses the services supported by Asset Repository Infrastructure. It includes:

- **The Asset Repository Manager** – we developed a web-based application embodying main functionalities for lifecycle management of assets and artefacts.
- **IDE plug-ins** – we developed rich clients consuming services from the Asset Repository service tier, which thus allows developers to manage assets from within their integrated development environment (IDE).

- **Other software components from the RAGE ecosystem**, such as the Ecosystem Portal (EP), which harvests assets and metadata through an OAI-PMH service provider from Asset Repository Service tier.

The Asset Repository services constitute an open interface for creating, modifying, deleting, and searching RAGE assets. They are realised on top of REST APIs, JSON, JSON-LD [22] and RDF, using Software as a Service (SaaS) model in the cloud. Based on the functionality exposed by these services, they can be grouped as:

- **Asset Access Services** defining an open interface for accessing assets within the RAGE Asset Repository allow for retrieving asset packages and metadata, and to 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 SPARQL for querying on asset metadata and SKOS taxonomies data represented as RDF triples.
- **Asset Management Services** defining an open interface for administering assets, including creating, modifying, and deleting, provide an abstract level of the operations, thus hiding the complexities of 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 [13] to classify and describe an asset in educational and gaming contexts. For representation and storing Asset Repository adopts SKOS standard [23].
- **Authentication and Authorization Services** provide access for organisational needs. These services are implemented on top of Fedora Authentication and Authorization framework [17].

7 Usage scenarios

In order to observe how the asset repository together with related client tools can support the asset developers and other users, and how effective and useful the services are, which it is offering, we have designed various usage scenarios. Also, asset developers and game developers have been involved for evaluating the functioning and usability of the repository. In this section we will present the scenarios, and in the next section will present the main conclusions based on the observations of real users.

To populate the repository with metadata we used four usage scenarios. The first scenario is publishing/updating a game asset through the web-based interface offered from the Asset Manager. The asset developer signs in, creates/selects an asset, enters/updates metadata and uploads artefacts or a packaged asset (see Figure 3).

The second scenario is publishing/updating a game asset from GitHub. The asset developer again should sign in the Asset Manager, creates/selects an asset, provides the GitHub repository identifier and credentials (if required). The files (artefacts) and metadata from GitHub are automatically harvested and published in the RAGE Asset

Repository (using the GitHub API [24]). The user should also supply the rest of the required metadata.

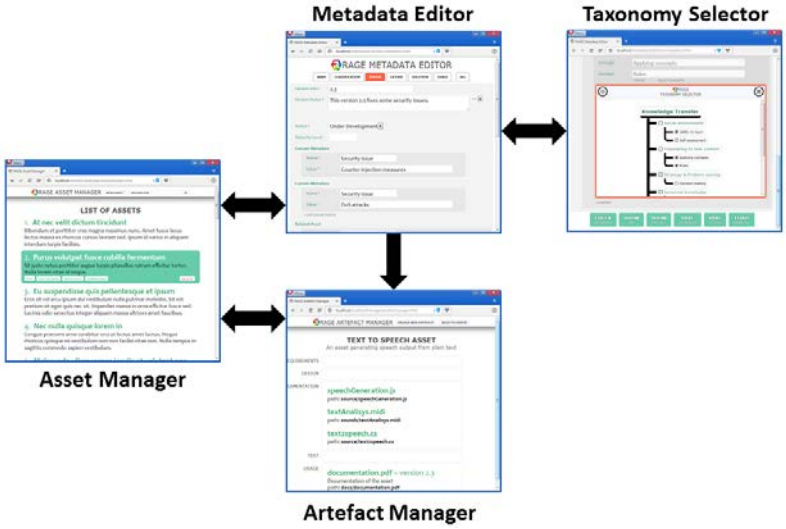

Figure 3. Using the RAGE Asset and Artefact managers, the RAGE Metadata editor and the RAGE Taxonomy selector to populate the repository

In the third scenario, we tested publishing/updating a game asset from an IDE. For this scenario we developed an Eclipse IDE plugin. The asset developer opens the asset project in the Eclipse IDE; using the plugin the developer creates/updates the asset in RAGE Asset Repository within the IDE, providing credentials and needed metadata.

The fourth scenario: Asset consumers can search for a game asset using full text or advanced search, browse the repository, view assets metadata and download assets or artefacts for reuse.

At the moment, the repository is populated with the metadata of 12 currently developed Assets in RAGE project.

8 Scenario evaluation

An evaluation of the usage scenarios was carried out by involving a group of 9 end users, viz. asset developers from the RAGE project. Preliminary findings of this user panel support the relevance of the repository system. Comments about the first version of the repository and related client tools can be summarized as follows:

- Users can easily work with basic services such as searching, downloading or uploading assets to the repository.
- Users need more specific instructions how to populate the repository with metadata.
- The metadata editor improved the process of populating the repository for users.

- Users encounter problems to identify the source of the information related to some of the metadata fields, like keywords and others.
- There is a need to automate further the definition of metadata fields.

While the evaluation is preliminary and relatively informal, the initial acceptance is positive, and confirms the viability of this first step within the RAGE Project.

9 Conclusions and future work

In this paper, we presented a unique software architecture supporting the lifecycle of reusable software components for applied gaming. The main innovation is related to the combination of RAGE Asset Model and RAGE Asset Metadata Model, backed up with server-side infrastructure (repository and services) and many end user tools. The software architecture plays a pivotal role within the RAGE Ecosystem, developed for the RAGE project and is considered of strategic importance for the domain of applied gaming.

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. It also 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. A first provisional launch of the repository integrated in the RAGE social platform is expected in 2017.

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. García Sánchez, R., Baalsrud Hauge, J., Fiucci, G., Rudnianski, M., Oliveira, M., Kyvsgaard Hansen, P., Riedel, J., Brown, D., Padrón-Nápoles, C.L., Arambarri Basanez, J.: Business Modelling and Implementation Report 2, GALA Network of Excellence, www.galanoe.eu.
2. Stewart, J., Bleumers, L., Van Looy, J., Mariën, I., All, A., Schurmans, D., Willaert, K., De Grove, F., Jacobs, A., Misuraca, G.: The Potential of Digital Games for Empowerment and Social Inclusion of Groups at Risk of Social and Economic Exclusion. Joint Research Centre, European Commission, Brussels. http://ftp.jrc.es/EURdoc/JRC78777.pdf (2013)
3. RAGE: Project Web site (2015) http://www.rageproject.eu .
4. Ackerman, L., Elder, P., Busch, C.V., Lopez-Mancisidor, A., Kimura, J., Balaji, N.A.: Strategic reuse with asset-based development, IBM RedBooks (2008) http://www.redbooks.ibm.com/redbooks/pdfs/sg247529.pdf
5. Kattau, S.: Atego launches RAS-based asset repository, SD Times Magazine, February 13, 2013, http://sdtimes.com/atego-launches-ras-based-asset-repository/#ixzz3wwMlvLJ8

6. Hong-min, R., Zhi-ying, Y., Jing-zhou, Z.: Design and Implementation of RAS-Based Open Source Software Repository, Proc. of te Sixth Int. Conf. on Fuzzy Systems and Knowledge Discovery, Vol.2, pp.219-223 (2009).
7. Moura, D. S.: Software Profile RAS: estendendo a padronização do Reusable Asset Specification e construindo um repositório de ativos, Master's thesis, Univ. Federal do Rio Grande do Sul, Brasil (2013) http://www.lume.ufrgs.br/handle/10183/87582
8. Hilliar, G.: Developing Cross-Platform Mobile Apps with HTML5 and Intel XDK, in Dr. Dobb's Journal, UBM plc. (2014)
9. Böhm, T., Klas, C.-P., Hemmje, M.: Supporting Collaborative Information Seeking and Searching in Distributed Environments. In Proc. Of the LWA 2013 Conference, Bamberg, Germany, pp 16-20 (2013).
10. Stefanov, K., Nikolov, R., Boytchev, P., Stefanova, E., Georgiev, A., Koychev, I., Nikolova, N., Grigorov, A.: Emerging Models and e-Infrastructures for Teacher Education, 2011 International Conference on Information Technology Based Higher Education and Training ITHET 2011, IEEE Catalog Number: CFP11578-CDR, ISBN: 978-1-4577-1671-3.
11. Dekkers, M.: Asset Description Metadata Schema (ADMS). W3C Working Group (2013)
12. Van der Vegt, G.W., Westera, W., Nyamsuren, N., Georgiev, A., Martinez Ortiz, I.: RAGE architecture for reusable serious gaming technology components, International Journal of Computer Games Technology, Vol 2016 (2016), http://dx.doi.org/10.1155/2016/5680526 .
13. A. Georgiev, A. Grigorov, B. Bontchev, P. Boytchev, K. Stefanov, K. Bahreini, E. Nyamsuren, W. van der Vegt, W. Westera, R. Prada, P. Hollins, P. Moreno. The RAGE Software Asset Model and Metadata Model, Serious Games, 2nd Joint Int. Conference, JCSG 2016, Springer, V. 9894 Lecture Notes in Computer Science, pp. 191-203, 2016.
14. Martin, James: Rapid Application Development, Macmillan, 1991.
15. Hollins, P. Westera,W. Manero Iglesias, B.: Amplifying applied game development and uptake, In Proceedings of 9th European Conference on Game-Based Learning ECGBL 2015, pp. 234-241, Steinkjer, Norway (2015)
16. Duval, E., Hodgins, W., Sutton, S., Weibel, S. L.: Metadata principles and practicalities. D-lib Magazine, 8(4), DOI: 10.1045/april2002-weibel (2002).
17. Woods, A.: Fedora 4.3 Documentation https://wiki.duraspace.org/display/FEDORA43/
18. Broekstra, J., Kampman, A., van Harmelen, F.: Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema. First International Semantic Web Conference, Lecture Notes in Computer Science, pp 54--68, Springer Verlag (2002).
19. Smiley, D., Pugh, E., Parisa, K., Mitchell, M.: Apache Solr 4 Enterprise Search Server, Packt Publishing, ISBN: 9781782161363 (2014).
20. Lagoze, C., Van de Sompel, H.: The Open Archives Initiative Protocol for Metadata Harvesting (2015) https://www.openarchives.org/OAI/openarchivesprotocol.html
21. SPARQL 1.1: SPARQL 1.1 Overview, W3C Recommendation (2013)
22. JSON-LD 1.0: A JSON-based Serialization for Linked Data, W3C Recommendation (2014)
23. SKOS: Simple Knowledge Organization System Reference, W3C Recommendation (2009)
24. GitHub API: GitHub Developer Guide (2016) https://developer.github.com/v3/\
25. M. B. Carvalho, F. Bellotti, R. Berta, A. De Gloria, G. Gazzarata, J. Hu, M. Kickmeier-Rust: A case study on Service-Oriented Architecture for Serious Games, Entertainment Computing 6(2015), pp. 1-10, DOI:10.1016/j.entcom.2014.11.001
26. A. Gloria, F. Bellotti, R. Berta, and E. Lavagnino, "Serious Games for Education and Training," International Journal of Serious Games, Vol. 1, No. 1, 2014, pp. 100-105, ISSN: 2384-8766