BUILDING A REFERENCE ARCHITECTURE FOR VIRTUAL RESEARCH ENVIRONMENTS

Keith G Jeffery[1], Carlo Meghini[2], Cesare Concordia[2] and Francesco Furfari[2]
[1]*Keith G Jeffery Consultants*
[2]*ISTI - CNR, Italy*

ABSTRACT

Virtual Research Environments (VREs) aim to support multidisciplinary research and collaboration between researchers, but face great challenges from data heterogeneity, user experience issues, and fast changes to datasets. This complicates research on multidisciplinary societal challenges. The objective of this paper is to present a reference architecture and components that can be (re)used to construct a general, domain-specific or cross-domain enhanced VRE (e-VRE) that enhances FAIR-ness of data and process interoperability across research infrastructures. The e-VRE makes it easier to conduct multidisciplinary research by assisting the researcher in accessing and utilising the assets of research, commonly stored in digital Research Infrastructures. The reference architecture also reduces complexity by interfacing with research management systems.

KEYWORDS

Virtual Research Environment, IS Design and Development, Distributed Computer Systems

1. INTRODUCTION

VRE4EIC (A Europe-wide Interoperable Virtual Research Environment to Empower Multidisciplinary Research Communities and Accelerate Innovation and Collaboration) is a three-year project funded by the European Commission in the context of the H2020 Program, under the topic EINFRA-9-2015 e-Infrastructures for virtual research environments. The major goals of the VRE4EIC project are the design of a Reference Architecture (RA) for VREs and the development of a Canonical Prototype that implements the RA. Generally speaking a Reference Architecture presents a description of a system in terms of its components and their relationships; it can be viewed as a sort of template that can be used to implement systems in a family of applicative domains. The components of the VRE4EIC RA have been designed to be used to build a general, domain-specific or cross-domain VREs that enhances FAIR-ness of data and process interoperability across research infrastructures. A VRE built using the VRE4EIC RA components is called *enhanced VRE* (e-VRE). The VRE4EIC RA has been designed to meet five main requirements:

- Increase VRE usability in different interdisciplinary domains by closely involving user communities and real-world use cases.
- Increase the quality of VRE user experiences by providing user centered, secure, privacy compliant, sustainable environments on searching data, composing workflows and tracking data publications.
- Increase the deployment of the VRE on different clusters of research infrastructures by abstracting and reusing building blocks and workflows from existing VREs, infrastructures and projects.
- Improve the contextual awareness and interoperability of the metadata across all layers of the resources in the VRE.

This paper presents the VRE4EIC RA, by first illustrating the methodology followed in defining the Reference Architecture and the derivation of a technical architecture from it.

2. VIRTUAL RESEARCH ENVIRONMENTS

Sharing research data is a critical activity to improve global collaboration on contemporary scientific research (Kim, 2017). A large diversity of datasets is already becoming available from various sources, including government data, geographical data, structured media data and life sciences data (Musto, Basile, Lops, de Gemmis, & Semeraro, 2017). Many research challenges span multiple scientific disciplines and both cross-domain and generic solutions need to be developed (Khan et al., 2017). Such solutions require the availability and integration of data, publications, software and other resources from multiple disciplines. VREs encourage multidisciplinary research by facilitating a research environment for the provision and (re)use of research data and other resources (Terras et al., 2016). They can be used to provide researchers with heterogeneous data, presented in a homogenous and user-friendly way. VREs provide researchers with access to research resources of a multiplicity of underlying e-research infrastructures minimizing much of the underlying complexity (Zuiderwijk, 2018).

3. DESIGN APPROACH AND METHODOLOGY

The development of a software architecture involves different points of views: the user, the system (the IT infrastructure), and the business goals. For each of these areas, the key scenarios/requirements should be identified as well as quality attributes. Then requirements should be refined into architectural functions and mapped into specific architectural components or modules. In the last 10 years different approaches for architecture specification have been proposed, largely inspired from the Software Development Life Cycle, SDLC [ISO/IEC 12207]. A typical SDLC includes different activities such as: understanding of business needs and constraints; elicitation and collection of requirements; functional architecture design; architecture design; implementation; testing; deployment; maintenance. The order in which these activities are to be executed is usually defined into a specific software development process such as Waterfall model, incremental model, RUP, V-model, iterative model, Agile model, Prototype model, to mention just a few. In the development of the VRE4EIC Reference Architecture, an incremental software development process largely inspired by the RUP process (RUP 2003) has been followed. In this Section, the main characterization of the process to the specific exigencies of the project constraints and activities are schematized. In particular, the Section provides details about activities concerning the software architecture specification and design that are: elicitation and collection of requirements; functional architecture design; architecture design. Elicitation and collection of requirements is a fundamental stage in the VRE4EIC RA, from a technical point of view, it involves different stages such as [ISO/IEC/IEEE 29148]:

1) High-level use case definition: a representative of the stakeholder community presents the business/mission drivers for the VRE4EIC RA considering also quality attributes, security aspects.

2) Use case definition: Stakeholders express scenarios representing their concerns about the system prioritizing when possible the main ones.

3) Specification VRE4EIC RA requirements: the main use cases are analysed and refined in more detail. The focus is on specifying what a system should do (the functional requirements) and on how the system should function (the non-functional, or quality, requirements) [ISO/IEC 25010, ISO/IEC 25030].

Figure 1 presents a class diagram where these entities are related to one another and to the entities subsequently derived in analysing the requirements to derive the Reference Architecture. The analysis of requirements is explained in the next Section, with reference to Figure 1.

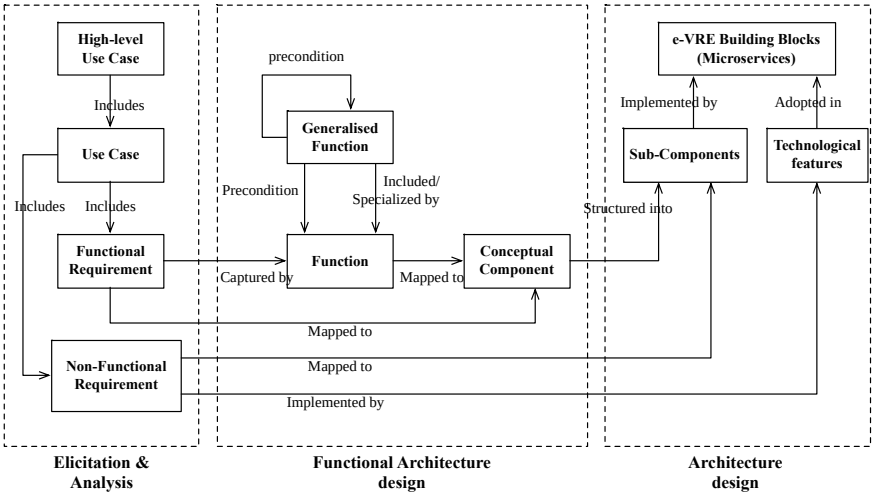

Figure 1. The Reference Architecture Derivation Process

The analysis started from the **Functional Requirements**. Each requirement has been considered individually, and the **Functions** required for its implementation have been derived. In order to ease the specification of functions, a set of **Generalised Functions** has also been derived, which are included or specialised by functions, or which may be used as preconditions by functions. During the execution of this phase, several design guidelines have been followed [RUP, https://msdn.microsoft.com/en-us/library/ee658124.aspx]. A synthesis of the most important ones is provided below.

- *Separation of functions*: Isolate from the requirements different functions with as little overlap in functionality as possible. The important factor is minimization of interaction points to achieve high cohesion and low coupling.
- *Aggregation of functions:* Identify possible generalization or composition relations between the isolated functions to improve the organization and readability of the functional architecture design.
- *Learn from similar projects:* analyze similar projects and documentation so to derive a high conceptual-level architecture description focusing mainly on high view of modules/components and communications and interactions between them.
- *Reduce Responsibility:* Assign to each component or module the responsibility for only a specific functionality or aggregation of cohesive functionality.
- *Minimal Knowledge:* Each component or module should be unaware of the internal details of other components.

This analysis of requirements into functions has been used for the assessment of the Reference Architecture. Overall, the approach has allowed maintaining the relationship between use cases, requirements and components, thereby realizing the traceability of the Reference Architecture. The **Components** that are required for the implementation of functions have finally been derived. The first step to derive components have been to refine and further decompose the Components into **Sub-Components** so that the identified functional and *non-functional* requirements are completely satisfied. Also for carrying out this step several design guidelines have been followed. Here below a synthesis of the most important ones is provided.

- Assess minimal knowledge: verify that each component does not rely on internal details of other components. In particular check that each component method is called from at least another object or component. Verify also that the method has information about how to process the request and, if appropriate, how to route it to appropriate subcomponents or other components.
- Avoid overloading of the functionality of a component: Avoid to overloaded components with many functions and applying the single responsibility and separation of concerns principles.
- Focus on communication between components: Understand the deployment scenarios and determine if all components will run within the same process, or if communication across physical or process boundaries must be supported—perhaps by implementing message-based interfaces.
- Define a clear contract for components: Components and modules should define a contract or interface specification that describes their usage and behavior clearly. The contract should describe

how other components can access the internal functionality of the component, module, or function; and the behavior of that functionality in terms of preconditions, postconditions, side effects, exceptions, performance characteristics, and other factors.

Components are detailed in next section.

4. THE VRE4EIC REFERENCE ARCHITECTURE

There is no common definition of a VRE system. Some scholars refer to the complete stack of software, data, computing resources, and sensors available to a researcher through a GUI. Others use the term VRE to refer only to an integrating 'hub', composed of a user interface component, a metadata catalogue and an integrating system (e.g., Zuiderwijk et al., 2016). In the second definition the integrating system assembles and provides components which access (via metadata) assets such as datasets, services, software components, workflows, computing facilities and sensors (we call these assets VRE resources). The Research Infrastructures (RIs) then provide access to the datasets, services and other resources. In this second definition, the 'hub' also assists the researcher in constructing and deploying workflows to meet the need, together with providing access to facilities for office functions, peer communication, publication, management of educational material and assistance with routine management tasks. Here we use the second definition because it reflects better the requirements of multidisciplinary researchers while also catering for domain-specific research. Considering the multi-tiers view approach, used in the design of distributed information systems (Schuldt, 2009), we can individuate three logical tiers in a VRE system:

- The *Application* tier, which provides to researchers and e-scientists functionalities to execute their experiments, to manage the system and to *expand* it, by plugging on it new components or tools.
- The *Interoperability* tier, which deals with interoperability aspects by providing functionalities for: i) enabling application tier components to discover and use VRE resources that may be stored in different locations and may be created according to different data models; ii) publishing VRE functionalities (e.g. using a Web Service API); and iii) enabling VRE applications and tools to interact with each other.
- The *Resource Access* tier, which implements actual integration functionalities, to enable VRE components to access RIs resources and services using the correct interaction protocol.

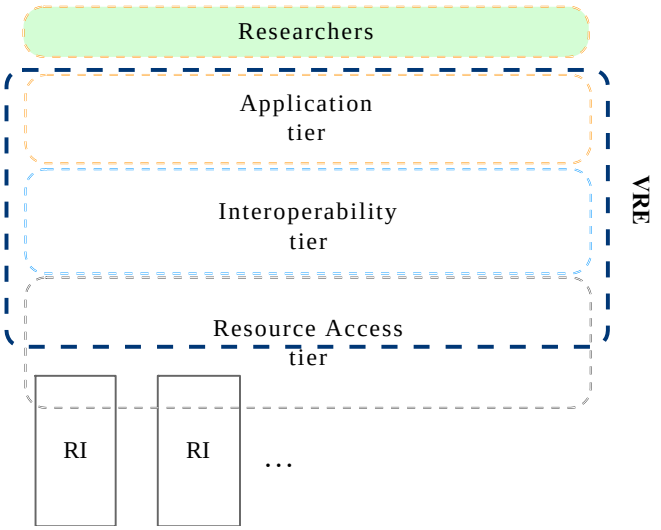

Figure 2. Architectural Tiers in a Virtual Research Environment

Figure 2 depicts the logical tiers of an VRE and shows their placement in an ideal space between the researchers that use the VRE and the RIs that provide the resources to the VRE. The VRE4EIC RA has been designed by distributing the implementation of the functional requirements in a number of conceptual

components and by logically *placing* each component in the tiers defined above. In particular the functional architecture design activity has identified six conceptual components:

- The RA management is implemented in the System Manager component. The System Manager can be viewed as the component enabling Users to use the core functionalities of an e-VRE: access, create and manage resource descriptions, query the e-VRE information space, configure the e-VRE, plug and deploy new tools in the e-VRE and more.
- The Workflow Manager enables users to create, execute and store scientific workflows and business processes.
- The Linked Data (LD) Manager is the component that publishes the information space using the LOD (Linked Open Data) paradigm, based on the RDF (Resource Description Framework) data model. Essentially this component provides functionalities to access the metadata concerning the e-VRE and the RIs in a form suitable for end-user browsing in a SM (Semantic Web)-enabled ecosystem.
- The Metadata Manager (MM) is the component responsible for storing and managing resource catalogues, user profiles, provenance information, preservation metadata used by all the components using extended entity-relational conceptual and object-relational logical representation for efficiency.
- The Interoperability Manager (IM) provides functionalities to implement interactions with RIs resources in a transparent way. It implements services and algorithms to enable the VRE to: communicate synchronously or asynchronously with RIs resources, query the RIs catalogues and storages, map the data models. The Interoperability Manager is also responsible for efficiently managing the integration of third-party software, enabling the e-VRE to virtually acquire any desired functionality that is not directly offered by any component.
- The Authentication, Authorization, Accounting Infrastructure (AAAI) component is the responsible for managing the security issues of the VRE system. It provides user authentication for the VRE and connected e-RIs, authorisation and accounting services, and data encryption layers for components that are accessible over potentially insecure networks. The AAAI component interfaces with external identity providers to enable single sign-on across the various connected infrastructures. For any authenticated user, it provides authorization services by using attributes provided by the external identity provider (if any). This includes user authorisations (role-based access) and accounting and billing of resources for which payment is required, both based on CERIF (euroCRIS, 2018) metadata provided by the metadata manager component.

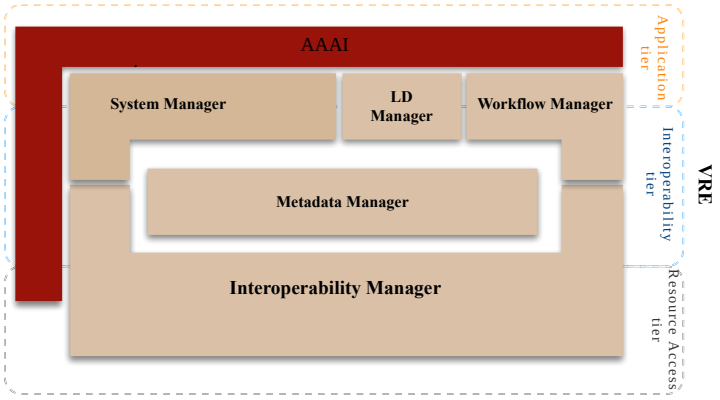

Figure 3. VRE4EIC RA Conceptual Components and VRE Logical Tiers

5. DERIVING A TECHNICAL ARCHITECTURE

The first step in designing the RA has been to refine the modularity of the system at functional level, by analysing functionalities implemented in every conceptual. The Figure 4 shows an overview of the sub-components in which every component has been structured, criteria adopted for this process has been

various. For instance, the Workflow Manager has been structured in three sub-components following the typical structure of a workflow engine: the Configurator that implements workflows definition functionalities, the executor implementing execution functionalities and the Repository implementing storage management. The same approach has been followed for LD Manager.

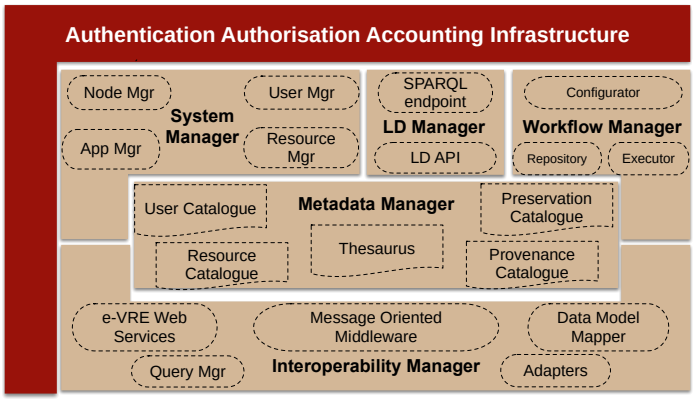

Figure 4. The designed Reference Architecture

The structure of the System Manager instead, depends from the consideration that managing a complex distributed system requires to deal with various kind of data and entities, for instance user *status* must be maintained across sessions, resources description data must be accessed, application life-cycle information must be monitored and managed, distributed configuration must be implemented etc. The functionalities of the System Manager has then been partitioned in four homogenous subsets and a subcomponent has been defined for every subset. A similar approach has been followed to define sub-components for the Interoperability Manager and for the Metadata Manager. The second step in the derivation of the technical architecture, has been to consider the FURPS+[1] (Ottinger, T., Langr, J. 2001) non-functional requirements, collected during the elicitation steps of the project and during the Gap Analysis, a survey conducted after the definition of the functional architecture to identify the most needed components in existing e-RIs and VREs. E-VRE Non-Functional Requirements can be synthesized as follows.

a) The e-VRE must be a dynamic system, it should be easily expandable by adding new software modules or replacing existing software components.

b) The e-VRE should reuse and integrate existing VRE tools, services, standardized components and workflows where appropriate, and develop new innovative ones where needed.

c) The e-VRE should be applicable to multidisciplinary domains, and it can be potentially used in every research domain.

d) The e-VRE functionalities should be exposed as services in a standardized way to enable developers to easily use them to develop new software that can be used in VREs.

e) The e-VRE must provide innovative standard software services to be *retro-fitted* to existing VREs to enhance them for their own domain purposes and for interoperability.

To implement these requirements, an approach based on Microservices[2] has been chosen. The two key concepts of Microservices architecture [Newman] *loose coupling* and *high cohesion* of components, perfectly respond to the above requirements. Every e-VRE microservice, called e-VRE building block, is an autonomous component that encapsulates al components that are needed for the implementation of its functionalities. Figure 5 shows the resulting micro-services, highlighting the functional components included in it, each characterized by the color relative to the tier where the component belongs.

[1] the FURPS+ acronym, devised by Robert Grady of HP, refers to the non -functional requirements categories named Functionality (Generality of Feature Set, reusability, security), Usability, Reliability, Performance, Supportability, and the constraints(+) on design, implementation, interface and physical properties, of the system.

[2] https://martinfowler.com/articles/microservices.html

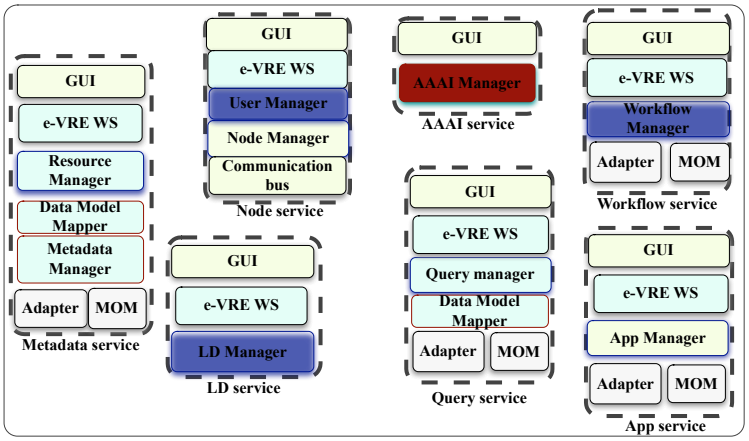

Figure 5. The Services in the Technical Architecture of e-VRE

As shown in the figure, changes has been made with respect to the structure of functional architecture. The main difference is that there is not a specific building block for the Interoperability Manager (IM). The IM is a crucial component to implement the cross domain capabilities of e-VRE, it is involved in every use-case where an e-VRE component needs to access or use assets (datasets, services, facilities, sensors etc) belonging to RIs. The IM sub-components can be divided according to the *integration level* they provide (Aloia N.; Concordia C.; Weinbach G 2002): ad-hoc Adapters, e-VRE Web Services and MOM component, provides integration at application level, Query Manager and the Data Model Mapper are used for integrating RIs at and at *data level*. A possible implementation of the IM could be to create a specific building block for the IM sub-components and call it from other building blocks. However, this approach will introduce dependencies when retro-fitting an e-VRE component or integrating distributed search functionalities into an existing VRE (requirements *e* and *f*). Application level integration sub-components (Adapters, MOM, e-VRE WS) have then been *distributed* in the various e-VRE Microservices, this means that these sub-components have been replicated and embedded into the building blocks using them. This solution helps developers in implementing requirements *d* and *e.*. Data level integration components (Query Manager, Data Model Mapper) have been implemented in a specific building block called the Query Service, this building block can be easily retrofitted in existing VREs and this implements requirements. In order to fully implement the non-functional requirements, the following technical solutions have been adopted in the e-VRE architecture:

- **Use event-driven communication to avoid component coupling.** The event-driven model is not blocking and is highly decoupled: every event is mapped into a *message*, messages are exchanged *asynchronously*. This approach removes direct dependencies between architecture components, and is a fundamental choice to implement the non-functional requirements listed above.
- **Using a distributed configuration service and synchronization service to manage coexistence of different endpoints**. An important feature provided by microservices architectures is that different services can be installed on different servers and also that different version of the same service can coexist on the same system. This has required the implementation of a distributed configuration service, synchronization service, and naming registry.

The assessment process has been performed by relating the VRE4EIC RA and vision to the ongoing work in the area of VRE, and then by validating the most significant Generalized Functions thereby showing the adequacy of the Reference Architecture from a functional point of view. The e-VRE building blocks are currently being developed, a repository has been created on GitHub to host the source code of e-VRE (VRE4EIC project, 2018). A live prototype of an e-VRE system is available[3], the main goal of the prototype is to collect feedbacks in order to individuate design errors and weaknesses of the reference architecture.

[3] http://v4e-hub.isti.cnr.it:8099/#!/login

6. CONCLUSIONS

The objective of this paper is to present a reference architecture and components that can be (re)used to construct a general, domain-specific or cross-domain enhanced VRE (e-VRE) that enhances the Findability, Accessibility, Interoperability and Reusability (FAIR) of data and process interoperability across research infrastructures. The purpose is to assist researchers and others in finding, accessing, interoperating and re-using data resources concerning global challenges such as global warming, climate change, pollution, food crises, refugees and natural disasters. The research problems behind those challenges are essentially interdisciplinary. Requirements for the e-VRE were elicited from 2VRE-related projects, 10 user interviews, a survey with 76 end-users and a characterization of 7 Research Infrastructures. A Reference Architecture for an enhanced Virtual Research Environment (e-VRE) was designed to particularly handle the above-mentioned requirements and limitations.

ACKNOWLEDGEMENT

The research leading to these results has received funding from the European Union Horizon 2020 Programme, Topic: e-Infrastructures for virtual research environments, Research and Innovation action VRE4EIC (A Europe-wide Interoperable Virtual Research Environment to Empower Multidisciplinary Research Communities and Accelerate Innovation and Collaboration), grant agreement n 676247.

REFERENCES

Aloia N.; Concordia C.; Weinbach G (2002) EAI: Concept and trends, ICSSEA 2002, Paris 2002.

euroCRIS. (2018). Main features of CERIF. Retrieved March 15, 2018, from https://www.eurocris.org/cerif/main-features-cerif

ISO/IEC 12207 International Organization for Standardization, "ISO/IEC/IEEE 12207:2008 - Systems and software engineering -- Software life cycle processes," ISO/IEC, Mar. 2008.

ISO/IEC 25010 International Organization for Standardization, "ISO/IEC 25010 - Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models," ISO/IEC, Mar. 2011.

ISO/IEC 25030 International Organization for Standardization, "ISO/IEC 25030 - Software engineering — Software product Quality Requirements and Evaluation (SQuaRE) — Quality requirements," ISO/IEC, June 2007.

ISO/IEC/IEEE 29148 International Organization for Standardization, "ISO/IEC /IEEE 29148:2011 - Systems and software engineering — Life cycle processes — Requirements engineering," ISO/IEC/IEEE, Nov. 2011.

Khan, S., Liu, X., Shakil, K. A., & Alam, M. (2017). A survey on scholarly data: From big data perspective. *Information Processing & Management, 53*(4), 923-944. doi: https://doi.org/10.1016/j.ipm.2017.03.006

Musto, C., Basile, P., Lops, P., de Gemmis, M., & Semeraro, G. (2017). Introducing linked open data in graph-based recommender systems. *Information Processing & Management, 53*(2), 405-435. doi: https://doi.org/10.1016/j.ipm.2016.12.003

Ottinger, T., Langr, J. (2011). Agile in a Flash.

RUP (2003) Rational Unified Process Best Practices for Software Development Teams, Rational Software White Paper TP026B, Rev 11/01, July 2003

Schuldt, H. (2009). Multi-Tier Architecture. In L. Liu & M. T. Ozsu (Eds.), *Encyclopedia of Database Systems*: Springer.

Sommerville, 2011. Software Engineering. 9th Edition, Addison-Wesley 2011

Terras, M., Warwick, C., & Ross, C. (2016). Building Useful Virtual Research Environments: The Need for User-led Design. In P. Dale, J. Beard & M. Holland (Eds.), *University Libraries and Digital Learning Environments* (pp. 151). London: Routledge.

Zuiderwijk, A., Jeffery, K., Bailo, D., & Yin, Y. (2016). *Using Open Research Data for Public Policy Making: Opportunities of Virtual Research Environments*. Paper presented at the Conference for E-Democracy and Open Government, Krems an der Donau, Austria.

Zuiderwijk, A. (2018). Analysing open data in virtual research environments: New collaboration opportunities to improve policy making. *The International Journal of Electronic Government Research, 13*(4), 76-92. doi: 10.4018/IJEGR.2017100105