Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 4 (2011) 627–636

Procedia
Computer Science

International Conference on Computational Science, ICCS 2011

A-R-E: The Author-Review-Execute Environment

Wolfgang Müller[a]*, Isabel Rojas[a], Andreas Eberhart[b], Peter Haase[b], Michael Schmidt[b]

[a]*SDBV group, HITS, 69118 Heidelberg, Germany*
[b]*fluid Operations,69190 Walldorf, Germany*

Abstract

The Author-Review-Execute (A-R-E) is an innovative concept to offer under a single principle and platform an environment to support the life cycle of an (executable) paper; namely the authoring of the paper, its submission, the reviewing process, the author's revisions, its publication, and finally the study (reading/interaction) of the paper as well as extensions (follow ups) of the paper. It combines Semantic Wiki technology, a resolver that solves links both between parts of documents to executable code or to data, an anonymizing component to support the authoring and reviewing tasks, and web services providing link perennity.

Keywords: Semantic Wiki, Linked Data, Extended links

1. Introduction

The main goal of an executable paper is to increase comprehension, reproducibility and sustainability of electronic publications. We take a data-driven, loosely coupled, and distributed approach to support the life cycle of an (executable) paper: authoring, reviewing, publication and study. In the A-R-E system the main objective is to provide an environment where a publication is a structured complex entity enriched with features that support further exploration of the facts and hypotheses stated in the paper, as well as related information from external sources.

We will present the concept and features of the A-R-E based on the types of users of the system, which we have defined as: author(s), reviewer(s), publisher and final reader(s). For each of these we consider (i) the desired user experience and (ii) the technical needs for the necessary functionality.

The **authors** of an executable paper require an environment that supports him or her in providing, enriching and linking content. It has to be simple and flexible; otherwise, no one will enhance the papers as needed. For increasing adoption, the authoring environment needs to preserve the freedom of choice of tools, e.g. the data analysis tools to be used. It is hard to imagine that scientists will commit to one platform that restricts the way that they do experiments or produce data, so the possibility to export and import text files and data is crucial. Linking information should be possible at different levels of details and to different information source types. The use of links from named entities and figures or tables into Semantic Web content (web-based databases, web-enabled

* Corresponding author. Tel.: +49-6221-533-231; fax: +49-6221-533-231.
E-mail address: wolfgang.mueller@h-its.org.

1877–0509 © 2011 Published by Elsevier Ltd. Selection and/or peer-review
under responsibility of Prof. Mitsuhisa Sato and Prof. Satoshi Matsuoka
doi:10.1016/j.procs.2011.04.066

articles, etc.) are necessary (now widely available, see Attwood *et al.* [1]) features that need to be supported. Furthermore, it would be desirable to allow the author to link sections (chunks) of information to sections within external (referenced) articles.

Apart from the authoring aspects *per se*, the system should support the author in submission as well as in the revision process (interaction with the reviewer). The author needs to be able to include additional information supporting the work presented in a paper, such as files with raw data from which a diagram is generated, or a program used to process the data, or even links to specific parts of a referenced publication (and not to the publication as a whole). Supporting theses needs would offer the authors an integrated platform for the management of their executable paper and its parts, covering the processes of writing, referencing, annotation, proofing, submission, and revision of the paper.

From the **reviewers'** point of view, the executable paper environment should facilitate the understanding and verification of the paper. The reviewer needs to have access to supplementary files such as data and executable code. Easy navigation and commenting of the paper's context using its structure is also a desired feature that helps the reviewer in his or her tasks. Referencing from one section to another allows the reviewer to concentrate on a certain aspect of the paper, for example, to follow-up on a given topic mentioned in the abstract, the author can link the information in the abstract to the related section, allowing the reviewer to use this link during the revision process. One of the main tasks in the reviewing process is the verification of the paper by consulting related work or information, which has to be supported by the system. The reviewer should be allowed to write his or her review using references to the content of the paper, to external references or to his or her own supplementary content or to the paper's supplementary files.

Often, resolving references means for a reader not only getting the paper B referenced, but also finding out which paragraph is providing the **information bit pertinent** to paper A. Similar considerations apply to figures in a paper. One would like to be able to navigate from data points in a plot to the data items in raw experimental data that led to these points, say by being pointed to an excel sheet column with experimental data. Furthermore, one would like to be able to navigate into the **program code** that led to a given aggregation of data. Evidently one will not be concerned in programming details, but rather in the implementation of the **main bits of code** leading to a plot, graph, or other figure. If given the chance, one would like to be able to explore the data further, either by changing the code applied or the data analyzed. It is important that the reviewing environment facilitates the communication reviewer-author and reviewer-publisher, all bit it in a secure and anonymous (for the case author-reviewer) manner.

Publishers need to control the anonymous peer review process, e.g. give the reviewers rights to view the document and supplementary data before acceptance and then open up the publication of the data as needed. By facilitating the interlinking among its papers the system can improves the quality of research as well as the repeatability of associated experiments.

The **final reader** can be viewed as a reviewer with limited authorization. The system should offer the same facilities as to the reviewer for the navigation and exploration of the paper, but these will be restricted by the access rights granted by the paper's authors to the supplementary files of the paper. If authorized by the publisher the system will allow the reader to make his or her personal "notes" on the paper, annotations and complementary information. In the last few years there have been multiple efforts towards augmenting the information provided by a publication, in order to facilitate its comprehension as well as to extend the knowledge it provides. Several Web-based tools allow the identification of terms in the paper against a set of ontologies or databases adding relevant hyperlinks to target pages. Attwood *et al.* [1] provides a broad and detailed overview of techniques and efforts.

Taking into account the features that we understand and think are required by the different types of users of an executable paper environment, the rest of this document will be structured as follows. First we will define our concept of the A-R-E system, highlighting the main goals that we aim to achieve. We then present the main components of the A-R-E system. After this we exemplify how an author and a reviewer could use the system and highlight also the role of each of the A-R-E components.

2. Concept

Our mission is to support data-driven navigation, analysis, visualization, and annotation of the publication in the different stages of the life-time of a paper: creation, revision, publication, analysis and extensions (follow-ups).

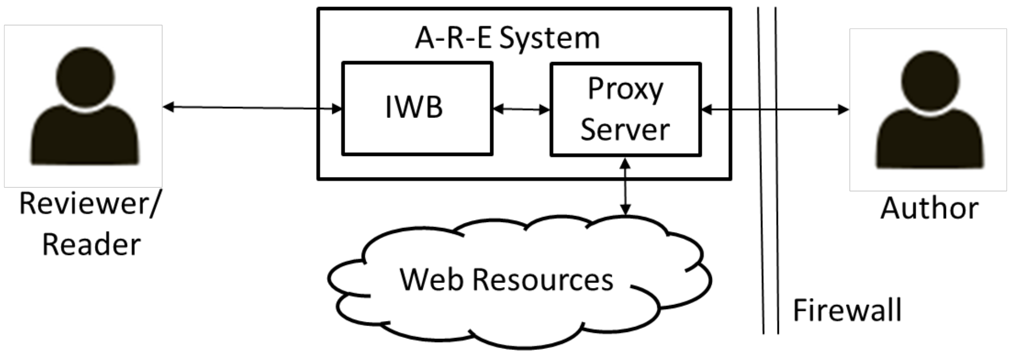


Figure 1: Architecture of the A-R-E system

We consider that the author of a document is often behind a firewall, is not always root on his/her machine and does not necessarily want to go through extended administrative motions just for sharing some data with a reviewer. Furthermore we need to respect the reviewer's anonymity during the review. Finally, we need to resolve URLs specifying document regions using an appropriate component. As a consequence, the A-R-E system consists of the Information Workbench (IWB) [7], a Semantic Wiki-based tool for authoring and linking content, plus a proxy server component that reside at the publisher's location. The proxy server takes care of anonymization and firewall circumvention. In addition there is a local component running at the location of the author's that interacts with the proxy server component. This base architecture is depicted in Fig. 1 and will be further explained below.

Figure 2 below illustrates a simplified diagram of a paper's life cycle, indicating for each state the transition actions and the type of user that executes it. Each state is normally comprised by more than one task, which in turn can be iterative, so there could be multiple cycles in the writing process before submitting the final draft of the paper.

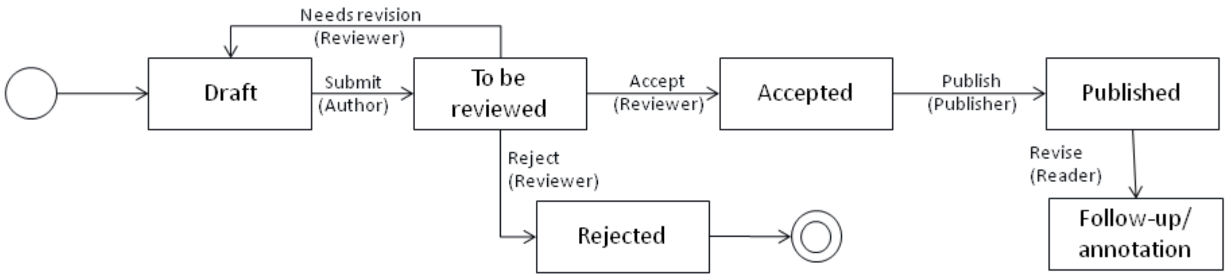


Figure 2: State flow diagram of the phases in the life cycle of the (executable) paper

Although the paper will go through different phases and probably suffer modifications or additions in these phases, we depart from a general intrinsic structure, which defines the system functionalities of the system. Figure 3 shows how an executable paper is represented in our system. At its core, the authors model the structure of their publication. Typically, it would be defined according to the sections and subsections contained in the paper, such as Abstract, Introduction, and other chapters. When importing a paper from an existing document (e.g., a Word file), the system could propose a structure according to the content of the document.

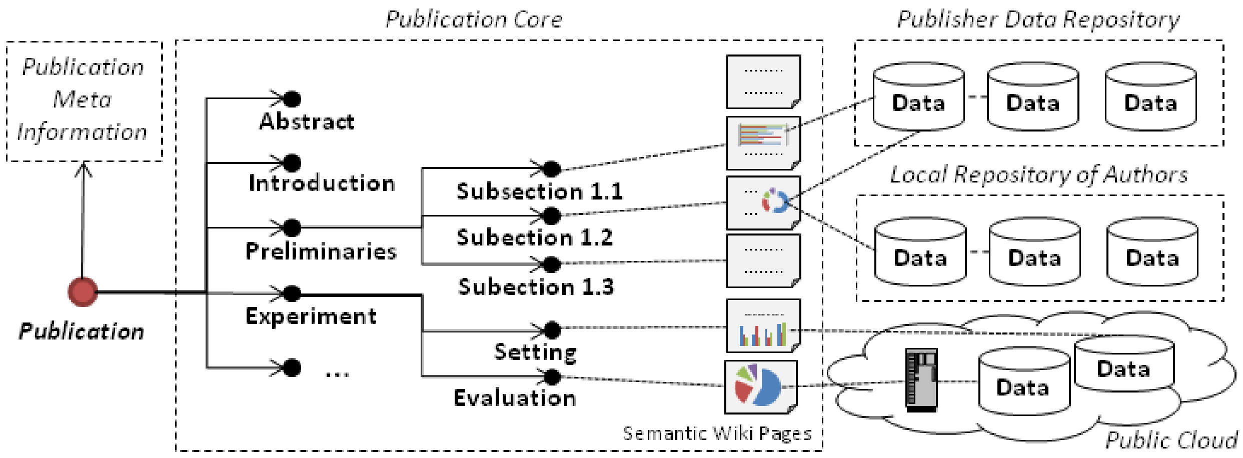


Figure 3: Structure of an executable paper in our A-R-E system

Apart from some basic constraints enforced by the publisher (like, e.g., the presence of an Abstract), the authors are totally free in defining the structure of their publication, and all sections/subsections are semantically linked to each other. Following the paradigms implemented in the Information Workbench, these semantic links are stored in the form of RDF data [8], the W3C standard for representing and exchanging semantic information. The semantic links connecting the components of the paper can be of benefit within the reviewing process, during publishing, and also once the paper has been published: given that parts of the paper are treated as first-class citizens in our model, the publisher could easily generate a table of contents, extract abstracts, or interlink related sections (even across different publications). As another example, once the paper has been published readers could annotate individual sections with comments, additional information, or related work.

As also shown in the figure, each section/subsection is associated with a Semantic Wiki page. These wiki pages can be collaboratively edited by the authors and, later on, processed by the reviewers. Such Semantic Wikis, which have recently gained attention not only in Semantic Web community (see e.g. [14]), differ from traditional wikis in that they allow to embed widgets that build upon an underlying semantic database, thus making it possible to create dynamic charts and dashboards that are filled according to the content contained in the underlying database. The Information Workbench offers built-in support to import data given in common formats such as tabular data, relational data, or RDF data. In addition, authors can choose to integrate both local data and data public data from global repositories. Coming with the proliferation of executable papers, we may also expect that it becomes common practice that authors publish associated data in the publisher's side, in a globally accessible data repository. Using the Information Workbench, the authors can then embed dynamic charts, dashboards, or other data visualization from public global data and local data directly into the Semantic Wiki pages, possibly combining data from multiple sources into a single dashboard. Furthermore, as another central feature authors can use the Semantic Wiki to put semantic links to executable code, which can later on be verified and run by the reviewers (we will discuss this issue in more detail later in this section).

We want to enable the linking of information in a paper to information within the paper as well as to external resources. These sources can be either data files or allow the processing of data. In addition to that, we want to enable the linking of parts of documents to parts of documents as an afterthought. Linking at the data level is the basis for the success of the Semantic Web. HTML allows linking from marked regions in documents to other documents and even anchors in documents. However, these anchors must have been prepared beforehand by the author of a document. In other words, the author of a document decides how the reader of a document is to read the document, which parts are to be referenced and which is the minimal granularity that can be referenced. Similarly, PDFs allow linking to pages or to named destinations that have to be created by the author.

This poverty in deep region-to-region linking possibilities is in stark contrast to the fact that there are quite some languages for specifying locations in general text, XML [9], spreadsheets, or even program code. For text, there are numerous examples such as the command language of the VI text editor (http://www.vim.org/). XPath [10] enables specifying regions in XML. In Spreadsheets, there is an implicit *de facto* standard across spreadsheets software (such as Excel, Open Office and others) on how to specify sheet regions in formulas. Aspect Oriented Programming [2] is about regions of code the so-called point-cuts to be affected by code changes, the so-called advices. All these region specification approaches can be used for enabling deep linking. Within our A-R-E concept, we are following a *pragmatic* way to enable deep linking without having to change or adapt web standards. We have chosen to enable linking via the use of a **proxy server/anonymizer** component which will be described in detail below. This notion of extended linking is complementary to efforts to join Excel and ontologies for improved use of Excel data [3,4,5].

Another important point that needs to be taken into account when defining the system is the provenance of information and tracking of revisions. This is a key factor to maintaining the integrity of an executable paper. The system implicitly will track changes and origin of information, check for lost links and modifications to referenced files. Here we plan to use techniques similar to those used in the SysMO-DB SEEK [6] (i.e. detecting file changes by generating, shipping and comparing cryptographic hashes).

Complementing the previous components, last but not least each publication has associated meta data. This meta data includes information such as the title, the authors, categorization information, keywords, associated proceeding information, etc. It is filled in by the authors using predefined forms when submitting the paper and will be aligned with common ontologies for publication meta data such as dublincore (http://dublincore.org/), to increase the reusability of the meta data description. Hence, as a major benefit, the publisher can directly publish all its meta data in a semantic data format such as RDF, to make it available to the scientific community. Related tasks such as the meta data annotation and data publishing processes are supported by the Information Workbench out-of-the-box.

Our conceptual view of the executable paper makes no reference to the location of the files (data, executables, or other supplementary files) that can be linked from the executable paper. The idea behind this is that this should be more or less seamless for the reviewers and readers, and controllable by the author. Inherently, the A-R-E is a distributed system: with distributed file management and distributed execution of tasks (all be it limited for the time being). In addition to the conceptual view of the executable paper, we need to cater for the concepts supporting the distributed nature of the system, namely *free choice of storage location*, *assuring author security and reviewer anonymity, and later virtual machines for execution.*

Free choice of storage location (data local, remote, or in the cloud): This functionality is closely linked to import functionality. The truth in modern science is that data can be at many places, the workstation at the desk, the computing center of the institution, the cloud, and other big data centers. Some of the data may be too big to ship by wire. Furthermore, we assume that most data analysis (e.g. on protein sequences, seismic data etc.) are carried out with specialized tools outside the system and that within the paper the author can reference the data, the tools, and the derived data, while these may reside outside the system.

An executable paper can refer to (or contain) the tools/programs that were used to process a given dataset or to obtain certain results. These elements (e.g. applications, data-sets, and results) can be stored in different locations, e.g. in the author's server, in the publisher's server, or in another (cloud) server. The system will provide the mechanisms to keep track of the referred elements and the relations between them (e.g. a certain file is the result of applying a certain tool/method on a certain set of data). Apart from defining his/her tools for data processing, authors, reviewers, and readers will also be able to apply (and refer to) tools and applications supplied by the system for a wide palette of data types, such as geographical data or protein data. Furthermore, authors will be able to create new widgets to incorporate (new) data analysis and processing tools into the A-R-E system, making them directly available to other users of the system.

Assuring author security and reviewer anonymity: Blind peer-review is still the prevalent way of evaluating papers. However, consider the following scenario: An executable paper has been submitted. As part of reading the executable paper the reviewer accesses a data file residing on a machine controlled by the author. Doing so, he leaves an IP address on the author's server. The IP address will allow the author to find out the reviewer's institution and (given a sufficiently small research domain) it will enable the author to find out the reviewer's identity. At the same time, in case of a hacking attack by the reviewer, the author would like to know who accessed his data and when. Both reader anonymity and author security are best achieved using an anonymizing proxy server under the control of the editor, as further described below.

Virtual machines for reproducibility: Providing a virtual machine that reproduces the conditions under which a certain application was ran goes beyond the scope of the first prototype that we aim to build. However, our prototype will provide the basis for the implementation of such features in future versions of our system. We will create the appropriate meta-data description in order to allow the author to specify the hardware and software requirements for the execution of the application, as well as meta-data on the result files describing the conditions under which these results were obtained.

3. Components of an A-R-E System

In this section we will describe the architecture of the system implemented to support the concepts described above. The A-R-E consists in principle of two main components, namely the **central authoring/reviewing/executing** environment and the **resolver/anonymizer.** In addition, there is a component that mainly works on the author's local computer (**author's local environment**) , which is normally behind a firewall in his or her organisation. The readers may want to access resources from outside the firewall, either at the editor's location or at the author's location or in distinct locations such as other servers or clouds.

The **central authoring/reviewing/executing** environment is maintained at the publisher's location. The **core component** of this environment is the Information Workbench (IWB [7]), a generic platform for building Linked Data applications, which has been developed by fluid Operations and is already productively used in fluid Operations' product portfolio. On demand, the user can also download a local copy of the Information Workbench, to author the paper in his or her local environment prior to uploading the content. The IWB comes with a built-in semantic database and provides full Semantic Wiki functionality, allowing users to author free-text sections, interlink such sections, and establish connections to integrated semantic data. Terms can be used as linking points within documents or document parts, and in turn these terms can be organized in graphs, supporting navigation and discovery by following paths of terms through the graph. The IWB design follows a self-service application development paradigm, *i.e.* it makes it easy to use and define widgets for searching, exploring, and processing data. In addition, it includes predefined components for the analysis and visualization of data (e.g. in the form of charts or dashboards) and supports the collaborative knowledge acquisition process, thus facilitating collaborative work on publications. With its built-in semantic database, it also makes it easy to attach meta information to executable papers, e.g. to categorize publications or to establish links between papers, authors, and conferences.

To build our A-R-E system on top of the Information Workbench we had to extend it by some novel features. Being designed as a platform for self-service application development, though, the IWB comes with APIs that allow to seamlessly integrate new modules and to couple it with other systems. Among the major changes was the support for editorial workflows in the Information Workbench, which enabled us to implement the authoring-reviewing-publishing process (cf. Figure 2 and the associated discussion). In addition, we had to integrate the IWB with the resolver/anonymizer component and added some new widgets supporting the submission and reviewing process. Most of the other tasks, like editing, metadata annotation, visualization support, or data export and import functionality could be realized out-of-the-box with features already in place.

The **author's local environment** refers to the machine residing at the author's location or to the machine on which the author stores his or her data or carries out the execution of his or her programs. The authors should be allowed to manage the files (of diverse nature) associated with the paper. It enables the following, distributed, functionality :

1. Authors are enabled to create locally executable papers, if they so wish, and then export their data for import into another instance of the Information Workbench.
2. Authors can provide (restricted) links to data that they do not want to upload, furthermore, they can provide the possibility to locally execute runs of software in their local environment.
3. Authors can control **multiple** local environments, e.g. the cloud for big data experiments and their local workstation for more iterative, less data-intensive tasks, like e.g. grouping results into plots.

4. The author must be enabled to share data with reviewers even before publication. Depending on the organisation setting up a restricted environment on the institute's server can incur severe hassle. Sharing data on the author's machine is often blocked by firewalls.

This means, that the author's local environment comprises an installation of the Information Workbench, as well as a tool that provides access to the data which are to remain outside the Information Workbench but should be shared from the author's machine. This component enables users to drag and drop data files into a shared area, to link files to each other and to create URLs that make the object accessible from the outside. The reader then can explore relations such as "resultfile A was generated from datafile B using matlab module M" and **access** the files in question. Obviously, this information is also sufficient in order to **run** analyses using input the shared data files yielding result files that can be shown to the reader/reviewer.

While the above caters for the authoring/reviewing/executing needs, we have not addressed the firewalls, yet. One possibility of circumventing firewalls involves a proxy server component. Let us consider as a general case a server A behind a firewall (that forbids incoming connections to A but allows outgoing connections from A), a proxy server B outside the firewall and a client C (behind another firewall that forbids C to run a server but allows C to build outgoing connections to servers) who wants to request data from A and is blocked by the firewall. As a consequence, C cannot be served by A directly. How to resolve the challenges of this scenario can be illustrated by using the interplay between the **author's local environment** and the **resolver/anonymizer.**

The **resolver/anonymizer** component has the following functions:
1. help **resolving** A-R-E URLs that designate document regions.
2. shield reviewer data requests from the data provider, thus acting as an **anonymizer**.
3. provide some **security** for the data provider, as in case of need the owner of the anonymizer can trace who accessed data via the anonymizer.
4. And it can also be useful as a proxy in a **firewall piercing** scenario.

For our example, we consider the hypothetical web locations author.org resolver.com, reviewer.org, and thirdparty.org. Imagine a document at http://publisher.com/executablePapers/document.jsp linking to http://thirdparty.org/paper.pdf, page 1, words 20 to 50, and to http://author.org/anotherPaper.pdf, page 4, words 41-57. Furthermore consider that author.org is hidden behind a firewall, so it cannot be accessed directly. For use in the A-R-E system one will wrap up these URLs such as:

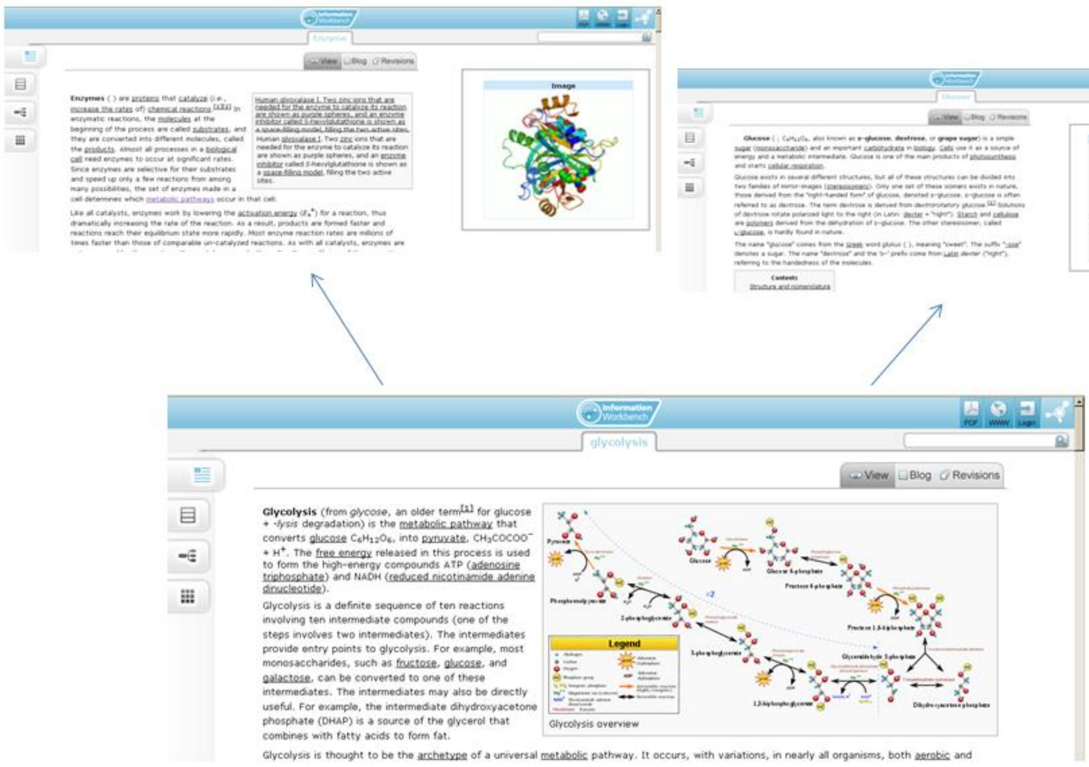
http://resolver.com/resolve?link="http://author.org/anotherPaper.pdf";page=4;words="41-57".

The following steps will be performed.
1. resolver.com receives the request. Before receiving the request, resolver.com has checked the requester's credentials and has made sure that the requester has the right to see the intended document. He has further determined *to which extent* the user can see the documents.
2. The local component on author.org periodically contacts **resolver.com.** As result from its latest poll, author.org learns that *anotherPaper.pdf* is requested
3. author.org sends *anotherPaper.pdf* to resolver.com.
4. resolver.com marks up the PDF of *anotherPaper.pdf*. In particular it highlights the words 41 through 57 on page 4. According to the rights of the requesting user, the corresponding representation is handed out. For example, we could imagine that some users are able to receive PDFs that allow cut-and-paste, others just receive GIFs that enable reading but not further processing.
5. resolver.com forwards the enriched postscript to reviewer.org

Note that in this five-step process (i) the link to a pdf region has been resolved, (ii) reviewer.org got the requested document, (iii) author.org did not learn who requested the document (iv) author.org learned the request by polling, i.e. without listening to a socket, thus avoiding the most frequent firewall restrictions.

As can be seen in the example, the resolver is a RESTful web service.

4. A Walk Through A-R-E's Features



Enzymes ( ) are proteins that catalyze (i.e., increase the rates of) chemical reactions.[1][2] In enzymatic reactions, the molecules at the beginning of the process are called substrates, and they are converted into different molecules, called the products. Almost all processes in a biological cell need enzymes to occur at significant rates. Since enzymes are selective for their substrates and speed up only a few reactions from among many possibilities, the set of enzymes made in a cell determines which metabolic pathways occur in that cell.

Human glyoxalase I. Two zinc ions that are needed for the enzyme to catalyze its reaction are shown as purple spheres, and an enzyme inhibitor called S-hexylglutathione is shown as a space-filling model, filling the two active sites.

Like all catalysts, enzymes work by lowering the activation energy ($E_a^{\ddagger}$) for a reaction, thus dramatically increasing the rate of the reaction. As a result, products are formed faster and reactions reach their equilibrium state more rapidly. Most enzyme reaction rates are millions of times faster than those of comparable un-catalyzed reactions. As with all catalysts, enzymes are

Glucose (; $C_6H_{12}O_6$, also known as **D-glucose**, **dextrose**, or **grape sugar**) is a simple sugar (monosaccharide) and an important carbohydrate in biology. Cells use it as a source of energy and a metabolic intermediate. Glucose is one of the main products of photosynthesis and starts cellular respiration.

Glucose exists in several different structures, but all of these structures can be divided into two families of mirror-images (stereoisomers). Only one set of these isomers exists in nature, those derived from the "right-handed form" of glucose, denoted D-glucose. D-glucose is often referred to as dextrose. The term dextrose is derived from dextrorotatory glucose.[1] Solutions of dextrose rotate polarized light to the right (in Latin: dexter = "right"). Starch and cellulose are polymers derived from the dehydration of D-glucose. The other stereoisomer, called L-glucose, is hardly found in nature.

The name "glucose" comes from the Greek word glukus ( ), meaning "sweet". The suffix "-ose" denotes a sugar. The name "dextrose" and the "D-" prefix come from Latin dexter ("right"), referring to the handedness of the molecules.

Contents
Structure and nomenclature

Glycolysis (from *glycose*, an older term[1] for glucose + *-lysis* degradation) is the metabolic pathway that converts glucose $C_6H_{12}O_6$, into pyruvate, $CH_3COCOO^- + H^+$. The free energy released in this process is used to form the high-energy compounds ATP (adenosine triphosphate) and NADH (reduced nicotinamide adenine dinucleotide).

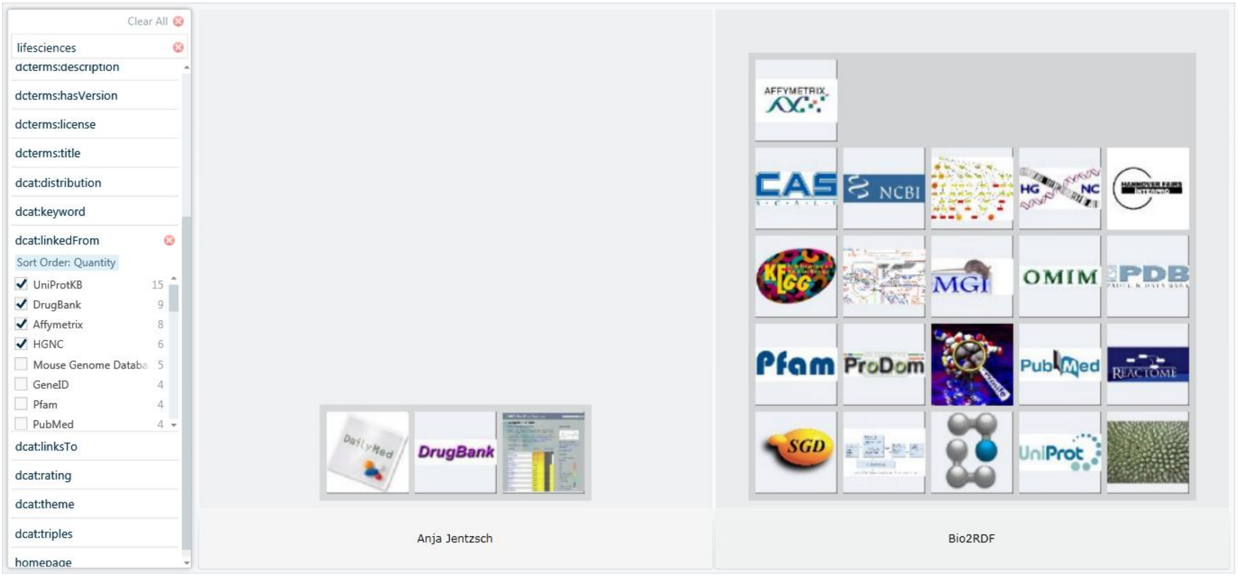
Glycolysis is a definite sequence of ten reactions involving ten intermediate compounds (one of the steps involves two intermediates). The intermediates provide entry points to glycolysis. For example, most monosaccharides, such as fructose, glucose, and galactose, can be converted to one of these intermediates. The intermediates may also be directly useful. For example, the intermediate dihydroxyacetone phosphate (DHAP) is a source of the glycerol that combines with fatty acids to form fat.

Glycolysis is thought to be the archetype of a universal metabolic pathway. It occurs, with variations, in nearly all organisms, both aerobic and

Figure 4: Representing linked content in the Information Workbench

To demonstrate the main features of the A-R-E system we have defined a case study scenario in the area of biochemical pathway analysis, where the HITS partner has experience in the processing of publications. However, in the demo version of the system we include some examples in other areas in order to demonstrate the generality of our solution and to present features that can be better/easier presented in different contexts. The **prototype** contemplates all the phases of an executable paper's life-cycle, namely the authoring of the paper, its submission, the reviewing process, the authors' revision, the publication, and finally the study (reading/interaction) of the paper as well as extensions (follow-ups) of the paper.

To write the publication the author can use a text processing program of his or her choice (this is not part of the system). The author can then import his or her draft of their paper into the system. The importing facility will recognize the main sections of the paper and define these as the structure of the paper, this can of course can be edited and modified by the author. The author can then use the A-R-E to explicitly reference related work (other executable and non-executable papers) and use the Semantic Wiki facilities to find and eventually link from the paper to external sources or to linked sources within the A-R-E system (See Figure 4).


Clear All
lifesciences
dcterms:description
dcterms:hasVersion
dcterms:license
dcterms:title
dcat:distribution
dcat:keyword
dcat:linkedFrom
Sort Order: Quantity

✓ UniProtKB	15
✓ DrugBank	9
✓ Affymetrix	8
✓ HGNC	6
☐ Mouse Genome Databa	5
☐ GeneID	4
☐ Pfam	4
☐ PubMed	4

dcat:linksTo
dcat:rating
dcat:theme
dcat:triples
homepage

Figure 5: Exploring and integrating existing life science data with the IWB

The A-R-E system already contains a wide pallet of life science databases from publicly available Linked Data repositories. This data can be explored and integrated by a single click of a button (see Figure 5). In our demo, we will show how the user can add new linked databases, such as the Systems Biology Ontology (SBO) [15] to make references from the article to terms in the SBO. The demo will then show the systems support to **reference** to the related data, **regions** in related data, program and application files which can be on the author's local server or imported into the publisher's server. We will show how the author can also use **applications** offered by the system, such as data analysis and visualization tools, and include the results into the paper. If desired, associated data can also be loaded directly into the A-R-E's internal database (or accessed dynamically from external Web sources), to be visualized on demand using internal charting and reporting widgets.

Once a paper is ready for **submission** the paper can be "proof executed" to guarantee its integrity (all links are functional and complete, references to external and system resources satisfy the necessary requirements, etc.). Once submitted the author will not be able to modify the paper. The paper will be made visible and executable (but not changeable) to the reviewers. Here the system will **support the reviewing process** by offering the search and linking facilities offered to the authors as well as allowing the application of analysis tools and other types of software to data supplied by the author or to other sets of paper. The reviewers can use the results of their analysis of the paper to create their reviews, referencing to the results they obtained and to parts of the paper.

To demonstrate the functionalies of the anonymizing proxy we will show a case where the reviewer needs to access data on the authors local environment. The author-reviewer cycle can be carried out several times (as many as allowed or necessary) and, assuming success, the paper will be ready for publication at some point. The management of the phases of the paper, once a paper has been registered in the system, is coordinated by the publisher. Finally, the **publication** of the executable paper then brings the paper into an environment similar to that of the reviewer. In this step, the publisher may limit the type of access to elements related to the paper, such as the code of applications, which could have been open for revision by the reviewers (possibly on request) but may not be accessible to the "general public". The potential reader of the paper benefits from the executable paper in several regards: improved search and exploration facilities may help to find relevant content and data more efficiently, precise links to parts of other documents facilitate the understanding of the paper and its relation to related work, executable parts of the paper could be reproduced and verified with only little effort, and data export functionality may give direct access to the data associated with the paper, allowing the reader to integrate the data into his/her own experiments. If desired, the publisher could choose to grant the author ongoing write access to the paper, allowing him to correct mistakes and extend previous results once the paper has been published.

5. Conclusions

We have developed the A-R-E- environment that implements our concept of an executable paper. Our prototype is based on a solid Semantic Wiki system augmented with a resolver/anonymizer component for the resolution of links amongst (parts) of documents, and for the control of security/anonymity issues involved in the reviewing process and in the sharing of data. Thirdly, there is a component (or multiple such components, where needed) at the author's location enabling simple through-firewall sharing, as well as performing execution of analysis on systems controlled by the author.

Together, these components are an environment that facilitates reading and comprehension of papers by making it easier to find what matters, to see how things fit together and to link them to outside sources of knowledge. We enable blending of established semantic web techniques with extended links that enable linking into regions of documents without creating anchors in these documents beforehand.

We have created a demonstrator to show the capabilities of the system, choosing as demo scenario the area of biochemical pathways. We see several possible extensions, including better support for virtual machines in order to execute papers in small-data scenarios at arbitrary locations and support for data streaming/multi-resolution approaches for big-data scenarios.

Acknowledgements
The authors wish to thank Dr. Ulrike Wittig, Renate Kania for discussion and the SysMO-DB and SysMO-LAB projects for ongoing fruitful collaboration.

References

1. T.K.Attwood, D.B.Kell, P. McDermott, J Marsh, S. R. Pettifer, D. Thorne: *Calling International Rescue: knowledge lost in literature and data landslide!* Biochem. J. (2009) 424, 317–333 (Printed in Great Britain) doi:10.1042/BJ20091474
2. G. Kiczales, J. Lamping, A. Mehdhekar, C. Maeda, C. V. Lopes. J. Loingtier, J. Irwin: *Aspect-Oriented Programming,* Proceedings of the European Conference on Object-Oriented Programming (ECOOP), Springer-Verlag LNCS 1241. June 1997
3. The ISA infrastructure http://isatab.sourceforge.net/isacreator.html
4. SysMO-DB RightField http://www.sysmo-db.org/rightfield
5. Anzo for Excel http://www.cambridgesemantics.com/products/anzo_for_excel
6. SysMO-DB project http://www.sysmo-db.org
7. The Information Workbench - Interacting with the Web of Data. P. Haase, A. Eberhart, S. Godelet, T. Mathäß, T. Tran, G. Ladwig, A. Wagner. Technical Report, fluid Operation & AIFB, October 2009. http://iwb.fluidops.com/.
8. RDF Primer. W3C Rec., Feb 10, 2004. http://www.w3.org/TR/rdf-syntax/.
9. XML (Extensible Markup Language). W3C, http://www.w3.org/XML/.
10. XPath (XML Path Language). W3C Rec. Nov 16, 1999. http://www.w3.org/TR/xpath/.
11. Vít Novácek, Siegfried Handschuh: Biomedical Publication Knowledge Acquisition, Processing and Dissemination with CORAAL. OTM Conferences (2) 2010: 1126-1144
12. Tudor Groza, Siegfried Handschuh, Georgeta Bordea: Towards automatic extraction of epistemic items from scientific publications. SAC 2010: 1341-1348
13. Vít Novácek, Tudor Groza, Siegfried Handschuh, Stefan Decker: CORAAL - Dive into publications, bathe in the knowledge. J. Web Sem. 8(2-3): 176-181 (2010)
14. M. Krötzsch, D. Vrandecic, M.Völkel: Semantic MediaWiki. International Semantic Web Conference 2006: 935-942
15. Le Novère N. (2006) Model storage, exchange and integration. BMC Neuroscience, 7(Suppl 1):S11. http://www.ebi.ac.uk/sbo/main/.