PhD Dissertation

International Doctorate School in Information and Communication Technologies

DISI – University of Trento

Universidad Católica
“Nuestra Señora de la Asunción”

Knowledge and Artifact Representation in the Scientific Lifecycle

Ronald Chenu-Abente Acosta

<u>Advisor</u>:

Prof. Fausto Giunchiglia

<u>Co-Advisor</u>:

Prof. Luca Cernuzzi

March 2012

node

node

Abstract

node

This thesis introduces SKOs (Scientific Knowledge Object) a specification for capturing the knowledge and artifacts that are produced by the scientific research processes. Aiming to address the current existing limitations of scientific production this specification is focused on reducing the work overhead of scientific creation, being composable and reusable, allow continuous evolution and facilitate collaboration and discovery among researchers. To do so it introduces four layers that capture different aspects of the scientific knowledge: content, meaning, ordering and visualization.

Executive Summary

node

Scientific papers or Articles were introduced during the 17th century when the first academic journals appeared. Since then these papers, along with the scientific conferences or symposiums, have become the cornerstones of the scientific community and research. Currently, not very unlike those early times, when an author wants to publish a scientific paper he has to submit a physical and or digital copy of it to an academic journal, where it goes through a process of Peer reviewing to determine if its publication is suitable (a similar process occurs in the case of submitting papers to conferences and workshops). Furthermore, the most common (and sometimes effective) way of discussing ideas with colleagues is still through the age-old tradition of organizing scientific conferences or symposiums.

This model based on papers, persons and scientific events has remained mostly undisturbed up to now. This is the case even if the Internet and the web are slowly reducing the costs related to the dissemination process and providing new ways of contact and interaction. Several studies have been carried out to find the existing limitations on the creation, dissemination, evaluation and credit attribution of these scientific artifacts. Furthermore, new types of less formal artifacts like web pages, blogs, comments, bookmark sites among others have also been increasingly popular in scientific environments. While these emerging web-based new types of scientific artifacts are generally less well-regarded than the traditional ones, it is nonetheless, irrefutable that they also are increasingly being used to disseminate, discuss, structure and, ultimately, advance scientific knowledge.

In the light of all the known problems and limitations of the current fragmented approach, explored in several studies, it is natural to wonder what would make an ideal scientific artifact format? One that is able to cope with the demands from creators, reviewers, publishers and consumers of scientific content while also enabling new interactions between these groups. More specifically, some of the requirements for this 'ideal' artifact format would be the following:

- *It is composable and complex* : allowing the definition of new artifacts from simpler ones, aggregating several types of data (e.g texts, spreadsheets, images, videos, etc) to facilitate the production of new knowledge or simply different executions or presentations of already existing knowledge.
- *It allows continuous evolution* : allows continuous creation and distribution of evolutions of the scientific artifact, while also supporting the current model of discrete releases at key points in time.
- *It facilitates collaborative work* : the previous properties like evolvability and composability, along with Internet-enabled interaction, greatly facilitate the creation of documents as collaborative effort between different actors.
- *It introduces improved models for certification and credit attribution* : provides easy-to-understand certification and credit attribution models with improved accuracy and that can be adopted gradually.
- *It is easy to find and classify* : making the large volume of information easier to filter and sort, by the use of metadata along with smart search algorithms, can be used to fight off the information overload.
- *It has a reduced work overhead* : provided suitable editor environments are created, it introduces no or very little additional work for the human actors.

Over the last years, a great variety of initiatives were developed to put these new technologies and interaction possibilities to the service of scientific discourse. The most relevant examples of these are:

- *General Models and Ontologies* : having an organized knowledge base that is specially constrained and adapted to the target domain (e.g., SWRC or Semantic Web for Research Communities) as the cornerstone of the approach would help to enable several of its proposed objectives; like providing well-foundedness and reusability at the conceptual level.
- *Semantic and Discourse Markup* : for the modeling and specification of scientific publications, general-purpose markup (like HTML, XML) and more focused approaches like (RDF and OWL) are extended into more specific formats like ABCDE, ScholOnto and SALT.
- *Collaborative Authoring Services* : content-oriented networks like Wikipedia enjoy a wide-spread appeal while focusing on the creation of knowledge artifacts through collaboration of a very large number of persons. Other approaches more focused on scientific research (like Swiki, Knol, Wikibooks and Ylvi) introduce modifications to the Wikipedia formula and thus become better platforms for the collaborative creation of scientific artifacts
- *Search engines* : a search engine specifically tailored for scientific papers, persons and events (e.g. Google Scholar).
- *Conference websites* : these sites contain announcements and concrete information about the venue and event of a conference. They are also mainly used to register (and pay) for participation (e.g. the ESWC2011 site)
- *Conference series websites* : these types of sites contain proceeding (i.e. collections of papers), give various announcements and offer special pages for the organizers to share information (e.g. the JCDL conference series site).
- *Submission management* : used to manage the submission process of papers to a conference (e.g. Easychair).
- *Social Networks* : like Facebook, almost unchallenged as a recreational and commercial social network. We would also want to offer scientific-related resources and content as the very center of the social network (e.g., Academia.edu).
- *Digital Library Portals* : like ACM's include several of the features we are interested in (albeit sometimes in a reduced or more basic manner).

The wide variety in the approaches and services mentioned shows that many of the functionalities and services that are considered interesting for scientific research are still somewhat dispersed. It would be normal for a researcher, for example, to first search for a paper in scientific search engine, then find it in the proceedings of a conference site and finally resort to email communication to comment or discuss. Besides the minor inconvenience of using multiple sites, the main problem is that no explicit tracking is kept over the whole process or even the interactions or discussions that are taking place. A platform used to capture and enrich the interactions that happen inside the research community has yet to appear, that is able to manage a large body of new and legacy papers while also for aggregating useful information and services related to persons and events from the community.

To bridge the seemingly antagonistic, conventional approach and the web-enabled new technologies, we propose a platform based on the SKO (Scientific Knowledge Object) specification; focused not only around the papers that contain the scientific knowledge or the persons that create it, but also around the events in which these persons meet to discuss this scientific knowledge. By using these three key elements, abstracted as metadata entities (i.e., a specific representation of objects in the real world) and enriched with semantic technology, we aim to accomplish the consistent aggregation of previously disjoint services; like provide access to conventional papers, provide a smart environment for creation and dissemination of new scientific artifacts, provide information to aid and enable visiting conference events and, offer services that would enrich the interactions and contributions with other researchers.

The SKO specification aims to represent and capture the knowledge from both the traditional and the new types of these scientific artifacts. The inspiration for this specification is mainly drawn from the comparison between the scientific artifacts and software, as several of the properties and processes currently used for software creation and management are deemed positive and interesting for the scientific publication world.
The current structural version of the SKO specification identifies four layers, with each of them used to capture a particular aspect of the scientific artifact :

1. *File layer* : this is the approach's foundational layer and the main connection to well-established and commonly-used content (e.g. text, pictures, tables).
2. *Semantic layer* : this layer includes all metadata attributes and relational information related the objects from the previous layer. Some examples of this include the title, authors, abstract and keywords of a paper.
3. *Serialization layer* : this layer is used for enabling and tracking aggregation and reuse of the information from the previous two layers. The ultimate objective of the serialization layer is to become the blueprint that is followed to order and choose the components from the graph-like content and knowledge repository (represented in the two previous layers), into a single linear document (much in the same way a software program's makefile compiles libraries and code into a running executable).
4. *Presentation layer* : by varying the information in this layer it is possible to create different styles of the same artifact (e.g. single column document, double column document, etc.) from the same basic serialization layer, while maintaining exactly the same data and knowledge.

This multi-layered approach is mainly aimed at enabling and facilitating the composition, reuse and collaborative creation of scientific artifacts. Furthermore; it provides the base for related works on improving the evolution, credit attribution, and search/navigation of these artifacts.

During the course of the PhD the concepts contained in the first two layers of the SKO specification (File and Semantic) were applied in the creation of a entity-based platform for knowledge production and dissemination. I particular, this platform aimed to create new ways for the members and interested people working in the Artificial Intelligence (AI) research community to interact; before, during and after their conferences. This platform, which was previewed during the International Joint Conference on Artificial Intelligence (IJCAI) 2011 and is still under development, allows the members of the community to discuss their work and to share their content (e.g. presentations, videos, notes and pictures) in a metadata-rich environment. Some of the end-user services the current version of the platform includes are:

- *Information display and navigation* : The main objective of this service is use the power of the semantic services to offer a seamless and effective entity navigation interface through a web interface.
- *Favorites and Personalized Entities* : As a way to assist the users remembering and following their personal interests, this feature allows to define 'favorite entities' (i.e., the papers, persons and events that have a special meaning or importance for the user)
- *Mobile Support* : Continuing with the objective of providing assistance to the members of the community, the mobile client of aims to offer key functionalities from the platform in a

ubiquitous manner. More specifically, it's objective is to provide support to manage the almost chaotic event of attending to a big scientific conference like IJCAI.

As a validation, the platform was previewed in a stand during IJCAI11 and interviews were carried out to obtain feedback from the attendants. The general results of these interviews were positive; the participants demonstrated great interest in the current version of the platform, while providing interesting suggestion for its future iterations.

The other mayor application of the SKO specification was the development of an ad-hoc creation environment prototype, aimed to implement and validate the concepts contained in the latter two layers (serialization and presentation). This prototype was used for creating a case study for reuse and aggregation of scientific information within the scope of the thesis itself. This means that most of the documents contained in the publications list (found on the next page) of this thesis and the thesis book itself were converted and written in the SKO specification using this prototype. A thesis self-evaluation postmortem on the use of this SKO-based creation environment is later given based on this creation environment. In this document the key advantages, disadvantages and future extensions of the SKO-based creation environment are discussed.

Finally, a smaller array of smaller applications and validation exercises were also development throughout the PhD period. These include the application of the SKO specification to other contexts (Events and Media), prototype and conceptual applications of the theory within the LiquidPub project [1] , an extension the SKO specification for the creation of discourse patterns in papers and, an exercise applied during a master-level course.

All these applications and validation exercises add up to suggest that the SKO specification, in fact, is able to adequately capture the scientific lifecycle and to offer useful services to the researchers that participate in it. Both main applications of the SKO specification (i.e., the metadata-enabled platform and the creation environment) will continue their development and evolution, potentially leading to creation several other applications.

Acknowledgements

This research has been funded by the LiquidPub project (Project number 213360, THEME 3: FP7-ICT-2007-C FET OPEN) http://project.liquidpub.org/ and supported by the organization of IJCAI11 http://ijcai-11.iiia.csic.es/

Special Thanks to Prof. Fausto Giunchiglia, Prof. Luca Cernuzzi and my loving family.

Contents

List of Figures

List of Tables

Part I. The Scientific Lifecycle

node

1. Introduction

node

Scientific papers or Articles were introduced during the 17th century when the first academic journals appeared. Since then these papers, along with the scientific conferences or symposiums, have become the cornerstones of the scientific community and research (as described in [1] and [2]).

Currently, as detailed in [3] and not very unlike those early times, when an author wants to publish a scientific paper he has to submit a physical and or digital copy of it to an academic journal, where it goes through a process of Peer reviewing to determine if its publication is suitable (a similar process occurs in the case of submitting papers to conferences and workshops). Furthermore, the most common (and sometimes effective) way of discussing ideas with colleagues is still through the age-old tradition of organizing scientific conferences or symposiums.

Furthermore, new types of less formal artifacts like web pages, blogs, comments, bookmark sites among others have also been increasingly popular in scientific environments. While these emerging web-based new types of scientific artifacts are generally less well-regarded than the traditional ones, it is nonetheless, irrefutable that they also are increasingly being used to disseminate, discuss, structure and, ultimately, advance scientific knowledge.

Over the last years, a great variety of initiatives were developed to put these new technologies and interaction possibilities to the service of scientific discourse. Some examples include scientific-oriented social networks(e.g., ResearchGate [2] , IamReasearcher [3] , Academia.edu [4]), digital library oriented sites (e.g., DBLP [5] , ACM Portal [6] , Mendeley [7]) and countless custom-made software oriented to specific conference-related functionalities (e.g., registration, submission systems, agenda management, mobile device support).

This divide and disconnection existing between the traditional and the emerging scientific processes and artifacts it is a focal point of this research, and will be further expanded and explained in the rest of this part of the thesis.

2. Problem Statement

node

Several studies have been carried out (for example [4] and [5]) to find the existing limitations on the creation, dissemination, evaluation and credit attribution of these scientific artifacts. The following are some of the most often-mentioned limitations:

- *Delayed creation and dissemination* : a lot of time is spent writing papers (which represent one of the main means for advancing scientific careers) instead of developing actual research. Furthermore, several months could go by between the submission of a paper and its publication. Thus hindering collaboration between the groups working independently on similar subjects.
- *Collaboration and reuse are not favored* : in scientific research, relying on previously existing work is currently only enabled by the aged reference and quotation systems. Meanwhile computer software, other types of artifacts that also encode knowledge, have found much better ways to reuse work and encourage collaboration while still maintaining clear credit attribution systems.

- *Flawed review and credit attribution methods* : peer to peer, the most widely used review model, is known to reject good papers that later go on to win awards of excellence and to approve papers that are later found out to be inaccurate or incorrect [6] . Furthermore, the current metrics used to evaluate works and authors encourages practices which are deemed as unhealthy for the progress of science.
- *Information overload* : the advent of web-based technology also introduced new challenges, most notably, an exponential rate of growth on the published artifacts. The aptly named 'information overload' problem described at [7] , significantly increased the difficulty of covering and keeping track of the latest development of a field.

As another widely used type of artifact, the software artifacts have also evolved over the years in an almost parallel fashion to scientific artifacts. Software artifacts refer to artifacts that contain encoded instructions for the use of a computer, thus despite also having the possibility of encoding knowledge they are not directly targeted at human consumption.

Despite each of them having their differences, their similarities are remarkable. For example, both cases involve collaborative work and exchange of ideas among project members and both normally need several iterations until they reach a final version. Nevertheless (perhaps needlessly) the treatment and evolution for both types of artifacts is very different, as shown in the following examples:

- *Development* : scientific knowledge artifact creation seems to be mostly still based on the Waterfall model, which makes it difficult to adapt to the changes and requirements that are frequently present in the research they represent. While, on the other hand; several other creation, development and distribution models are available and widely used for software artifacts. Creation models, such as extreme programming, and other agile software development models could be also applied to the production management of scientific knowledge as an attempt to bridge several of the delay and collaboration issues existing in the current approach.
- *Distribution* : when considering the great variety of distribution models currently available for software artifacts, it seems like the (still mainly peer-reviewed) scientific artifact distribution has lagged behind in its evolution. A clear example of this is the software-assisted distribution of artifacts through social networks. These artifacts can be created/modified and distributed at the same time (in fact the one is directly related to the other), which is a notable change from previous approaches.
- *Versions* : version control is normally used on software artifacts to enable collaboration or manage their contents. Furthermore, version control is also becoming increasingly used on non-software artifacts and is now commonly present on all the major document edition software. Nevertheless, the concepts of fork and branching (separating from a main version to diverge from it) and incremental releases, that may become useful in the scientific artifacts, are still used almost exclusively in software artifacts.

The following chapters survey how scientific and other related artifacts are created, evolved, disseminated, shared and accessed through the use of current technology. Several of the desirable qualities that the project would like to apply to the artifacts and its processes (evolutionary, collaborative, multi-faceted, among others) where also covered in detail. However, no single approach combining to all the requirements from the project was found between them.

This thesis proposes a new representation for scientific artifacts, the knowledge contained in them and the different scientific processes related to its creation. The main objective of this representation is to apply the interesting qualities of the software-artifacts and the new services brought by the advent of the web technologies to help bridge the divide between conventional/emerging science trends, while also aiming to diminish the limitations of the current scientific process.

3. State of the Art

node

The aim of this chapter is to draw inspiration from the achievements and ideas of ongoing works related to scientific artifacts .As such, a short review of current existing work related to the creation, evolution, collaboration and dissemination of scientific artifacts will be presented in the following paragraphs.

1. Software Artifacts

node

Software artifacts refer to artifacts that contain encoded instructions for the use of a computer, thus despite also having the possibility of encoding knowledge they are not directly targeted at humans. The following are some of the key factors and ideas from the software world that are considered interesting models for improving the creation and distribution of other knowledge artifacts.

Software Development

node

Software artifacts refer to artifacts that contain encoded instructions for the use of a computer, thus despite also having the possibility of encoding knowledge they are not directly targeted at humans. The following are some of the key factors and ideas from the software world that are considered interesting models for improving the creation and distribution of other knowledge artifacts.

Version control systems

node

Also called Versioning or Revision Control Systems (RCS), these refer to the management of multiple revisions of the same unit of information.RCSs are normally used to enable coordination of work between a group of people or manage the contents or the overall structure [9] resulting from the association of artifacts.

Artifacts are normally continuously rewritten and updated as new ideas and contributions are included in them. This process takes the form of a sequence of iterations applied to the artifact, forming a chain of transitions that contains the evolution of such artifact from its first draft to its final version. Some of these transitions could involve, and update or completion of a previous existing part, recovering ideas from past versions or even the merging of different contributions.

Furthermore, RCS has also been used on non-software artifacts like text editors [10] and are commonly present, in some form, on all the major document edition software.

The following are the major components that form part of RCSs:

- *Code repository* : a dedicated place where the software stores the entire project code base. This data could reside on the same machine where the development takes place, or on a remote system, accessed via network connections.
- *Files* : RCSs are about files and their evolution. As such, RCSs need to be able to handle and tell the differences between any kind of file.
- *Patches* : also called change-sets contain the differences between two versions and also the way of converting the earlier version into the newer one.
- *Locking* : used to manage the access to the files under version control and enable concurrent access to them and ensure their consistency.

A centralized RCS used for development works perfectly if it is assumed that all the interested users can contact the server at any given time. However this centralization also introduces a single point of failure, as any problems with the central server potentially denies its services to all users at the same time even if these users are somehow able to communicate directly.

As a response to this Distributed Revision Control Systems (DRCSs) were, somewhat recently, introduced. As their main feature DRCSs do not rely on a central repository holding the whole code-base but instead provides each developer with his/her own local repository.

Though this allows each user a great deal of autonomy, as they can branch and fork and update their copy as they see fit, it also introduces some additional considerations about the concurrency and consistency of the repository [11] . Well-known examples of DRCSs include Bazaar [8] , Mercurial [9] and Git [10] .

Summing up, revision control systems try to tackle two fundamental problems which affect every software development project:

1. Let different developers work on the same project while maintaining one single, coherent code base, thus avoiding the project code to become an inconspicuous mix of reciprocally incompatible contributions;
2. Keep track of every change made to the code base by developers, thus allowing editors to roll back whatever changes they did to the code. This also means that every change bears the name of her/his author, as well as the timestamp on which it was made.

Online Software Development Services

node

DRCSs also enable large scale and asynchronous coordination of collaboration within a group which, combined with easy of access and openness of the web environment is used to create development services communities that develop software using permissive and open source [12] licenses. Popular on-line open source development :

- *SourceForge* [11] , which has been targeted by several studies like [13] .
- *Ohloh* [12] , which auto-crawls the projects' repository to present activity and other data.
- *Launchpad* [13] , which is currently used by the Ubuntu Linux project among others.
- *Google code* [14] , which contains open source code based on the Google public APIs for developers interested in Google-related development.

2. Data and Knowledge Management

node

The terms data, information and knowledge are frequently used for overlapping concepts. Data is the lowest level of abstraction, which is the signal without interpretation and processing we touch by the minute. Information is the next level, which is endowed by meaningful data. And finally, knowledge is the highest level among all three that can always be used to generate new information. For knowledge management, there are three main research areas including knowledge retrieval, knowledge representation and knowledge application. This section will introduce a number of prevalent theories and applications related to data and knowledge management and, more specifically, the development of semantic-based technologies.

Metadata

node

Metadata [15] is 'data about data' or 'information about data', which is used to facilitate the understanding, characteristics, and management usage of data.
For instance, metadata would document data about data elements or attributes (name, size, data type, etc.), and data about records or data structures (length, fields, columns, etc) and data about data (where it is located, how it is associated, ownership, etc.). Metadata may also consist of descriptive information about the context, quality and condition, or characteristics of the data. A metadata set consists of elements in which each element refers to a label that describes particular information about a resource. A resource here is defined as 'anything that has an identity'. In the context of the Document-like Information Objects, a resource could be, for example, a document, monograph, web page, project report, or even people.

There are several initiatives and applications which aim to encompass issues of semantic standards in the domain of knowledge transaction with respect to description, resource discovery, interoperability and metadata exchange for different types of information resources.

- Dublin Core Metadata Initiative [16] (DCMI) - The Dublin Core Metadata Initiative is an open organization engaged in the development of interoperable online metadata standards that support a broad range of purposes and business models. DCMI's activities include work on architecture and modelling, discussions and collaborative work in DCMI Communities and DCMI Task Groups, annual conferences and workshops, standards liaison, and educational efforts to promote widespread acceptance of metadata standards and practices.
- Friend of a Friend [17] (FOAF) - The Friend of a Friend (FOAF) project is creating a Web of machine-readable pages describing people, the links between them and the things they create and do; it is a contribution to the linked information system known as theWeb. FOAF defines an open, decentralized technology for connecting social Web sites, and the people they describe.
- Open Archives Initiative Protocol for Metadata Harvesting [18] (OAI-PMH) - It defines a mechanism for data providers to expose their metadata. This protocol suggests that individual archives map their metadata to the Dublin Core, a simple and common metadata set for this purpose.

Hypertext

Hypertext is text which is displayed on a computer with hyperlinks [19] to other text or objects that the reader can access directly by a mouse click. It may contain not only text, but also tables, pictures, and other multimedia materials. Hypertext documents can be either static or dynamic. Herein, 'static' means the hyperlinks and the linked data are stored and prepared in advance, while the 'dynamic' one provides a dynamic response according to user's input and request. Nowadays, static hypertext is widely used for cross-reference collections of data in scientific papers and books.

A lot of work has been done in 1980's and 1990's aiming at content linking and reuse. In 1980, Tim Berners Lee created the earliest hypertext database system 'ENQUIRE'. Later, he invented the World Wide Web to meet the demand for document sharing among scientists working in different places all over the world. In 1992, the first Internet browser 'Lynx' was born. It provided hypertext links within documents which could reach into documents anywhere on the Internet. HyperCard was one of the most famous hypertext systems in 1980's and 1990's. From the year of 1987 to 1992, HyperCard was sold with Apple computer as promotions. Thereafter, DHM(Aarhus University, Denmark), Chimera(University of California, US), and Microcosm(Southampton University, UK) became dominant open hypertext system in those days.

In hypertext systems, a typed link is a link to another document or a document part that also carries the information about the link type. For instance, rather than merely pointing to the existence of a document, a typed link might also specify the relationship between subject and object, such as 'next' relation, 'previous' relation and so on. This facilitates a user to take actions like searching certain types of links or displaying them differently. In 'HTML 4.01 Specification', W3C predefined a set of link types which are as follows:

- alternate
- stylesheet
- start
- next
- prev
- contents
- index
- glossary
- copyright
- chapter
- section
- subsection
- appendix
- help
- bookmark

SGML(Standard Generalized Markup Language) is development of hypertext technology. It is also an ISO standard aiming to define generalized markup languages for documents. HTML and XML are both derivatives of SGML, while XML is a subset of SGML and HTML is an application of SGML. Some work has been done on content reuse, validation and source attribution as SGML/XML applications especially in the publishing area.
EDGAR [20] (Electronic Data-Gathering, Analysis, and Retrieval) system provides automatic collection, validation, indexing, acceptance, and forwarding of submissions. Authorized companies and other organizations can use it to require file data and information forms from the US Securities

and Exchange Commission (SEC).
DocBook [21] is a semantic markup language originally invented as an SGML application, designed for writing technical documentation related to computer software and hardware. It currently is an XML application. DocBook helps users create and reuse document content in a presentation-neutral form that can capture the logical structure of the document. After that, the content can be published in various formats, including HTML, XHTML, EPUB, PDF, etc., without asking users to make any changes to the source attribution.

In general, Hypertext research promotes content reuse, validation, and source attribution. Thus, it is the base for the definition of creation environment for authoring, annotating, search, and identification of types and patterns from scientific publications.

Markup Language

node

A markup language [22] is an artificial language using a set of annotations to describe the information regarding the structure of text or how it is to be displayed, which has been popularly used in computer typesetting and word-processing systems. Within a markup language, there is metadata, markup and data content. The metadata describes characteristics about the data, while the markup identifies the specific type of data content and acts as a container for that document instance. A well-known example of a markup language in use today in metadata processing is HTML, one of the most used in the World Wide Web. Furthermore, The Extensible Markup Language (XML) [23] is a general-purpose specification for creating custom markup languages. It is classified as an extensible language, because it allows the user to define the markup elements. XML's purpose is to aid information systems in sharing structured data, especially via the Internet, to encode documents, and to serialize data. Similarly, LaTeX is also a well-know example of a markup language but used mainly within the domain of documents and typesetting.

The Resource Description Framework (RDF) is a family of World Wide Web Consortium (W3C) specifications [24] , originally designed as a metadata data model. It has come to be used as a general method for conceptual description or modelling of information that is implemented in web resources; using a variety of syntax formats.

The Web Ontology Language (OWL) [25] is a family of knowledge representation languages for authoring ontologies, and is endorsed by the World Wide Web Consortium. This family of languages is based on two (largely, but not entirely, compatible) semantics: OWL DL and OWL Lite semantics are based on Description Logics [14] , which have attractive and well-understood computational properties, while OWL Full uses a novel semantic model intended to provide compatibility with RDF Schema. OWL ontologies are most commonly serialized using RDF/XML syntax. OWL is considered one of the fundamental technologies underpinning the Semantic Web, and has attracted both academic and commercial interest.

Lemon8-XML [26] is an effort of Public Knowledge Project (PKP) led by University of British Columbia and Stanford University. It's designed to help editors and authors convert scientific papers from typical editing formats such as MS-Word and OpenOffice, into XML-based publishing layout formats. It provides the ability to edit document and its metadata as well. Lemon8-XML is a stand-alone system that serves publishing processes more generally.

Document Management

node

Traditional document repositories based on hierarchical organizations such as filesystems, offer a limited view of a document space. Bush envisioned in 1945, some of the principles of that apply to modern document management systems; in which annotations, tagging and search are important features. In his work [15] , Bush motivates the use of associations in his hypothetical device memex instead of indexes as they are closer to the way human mind works. Some tools were developed to fulfil the vision behind memex [16] .

The placeless document management system [17] , or Placeless for short, is based on document properties rather than location in order to provide document organization. It extends from the normal document properties (i.e., creator, creation date, etc.) to offer universal properties (related to the base document) and personal properties (associated to document references). A key point of this proposal is the possibility of associating functionality to documents by means of active properties (i.e., properties with code associated). Thus, documents can raise actions when these properties are set (e.g., setting collaborator='John Doe' can send a mail to the collaborator).
Another important feature is the separation of content (physical data) from the document management. This allows reusing existing repositories and document processing tools.

Defining document workflows or lifecycles is another aspect of document systems. In some cases, the processes involving the document evolution are pre-defined (for example, a document system with focus on digitalization). In most advanced systems instead, a workflow or lifecycle facility is provided.

A document workflow approach based on Placeless is introduced in [18] . In this, active properties are used to provide the workflow functionality. Moreover, as with most real world scenarios, flexibility is also important in this context. A framework for document-driven workflows was proposed in [19] , which requires no explicit control flow. In this approach, the boundary of the flexibility is described by the dependency among documents (in terms of inputs and outputs). Nevertheless, as workflow operations are associated to changes in the documents, these changes must be done under the control of the workflow.

In a more general approach introduced in [20] , the processing of artifacts, from the creation to completion and archiving, is captured by lifecycles. Nonetheless, the flexibility offered is more focused on the artifact representation rather than lifecycle evolution and execution.

Besides providing organization, annotations and search capabilities and workflow management; document management systems provide usually versioning, sharing, document processing, and publishing.

A new scenario arises as new sophisticated tools are developed using the Web 2.0 model. Services such as Google Docs, Zoho Docs and social sharing systems are gaining more and more users due to their sharing and collaboration features. This definitely imposes new requirements as data becomes disperse, personal workspaces intersect with organizational workspaces, new devices and publishing platforms become available. Thus, interesting conceptual and design problems emerge in this ongoing area of research.

3. Collaborative Approaches

node

This section discuses the role of the ICT technologies in bridging the distances and enabling the collaboration of the creation and evolution of artifacts.

Online Social Networks

node

Social web services are based on communities of people that are brought together by the use of services like e-mail, forums among others. Enabled by these services social networks are formed [21] and, as these networks continue to grow in specific ways and patterns [22] , their users normally collaborate in the creation of several types of artifacts among other activities. Artifacts created within social networks include the following particularities:

- Artifacts over the social networks are created/modified and distributed at the same time (in fact the one is directly related to the other), which is a notable change from previous approaches.
- In most of the cases there is not a fixed end for the artifact's evolution, which is beneficial in the sense that it keeps the artifact updated but it may also cause its accuracy to be compromised.
- A relatively large amount of people has access to commenting, reviewing and sometimes even directly modifying the artifact.
- Having this large number of individuals collaborate is the key detail of the approach. This collaboration normally does not take into consideration the credentials of the persons and the sheer volume people collaborating can make the credit attribution hard to do accurately.
- Offers additional services like live references, classificatory tools, functionalities that allow the composition of parts and sections, topic-based recommendations, among other.

Depending on the nature of the social network the motivation for the creation of these artifacts also varies. On more communication-oriented networks like [27] users produce artifacts like photo albums to characterize people or places.
On the other more content-oriented social networks like Wikipedia [28] focus on the creation of knowledge artifacts through collaboration of a very large number of persons. While Wikipedia has been criticized because its model favours consensus and popular information over accuracy and its restriction of dealing only with well-known subjects (which disallows research), other similar approaches like Swiki [29] , Knol [30] and Ylvi [31] , introduce modifications to Wikipedia's formula to deal with these limitations.

The following are some examples of scientific-oriented social networks: ResearchGate [2] , IamReasearcher [3] and Academia.edu [4]).

Computer Supported Cooperative Work

node

While the previously discussed social web applies in general to all activities including ludic or other social interactions, social software within the context of work or creative processes is generally known as Groupware, or Collaborative Software.

Groupware forms the basis of Computer Supported Cooperative Work (CSCW), whose main objective is to study the technology to support people in their work [23] addressing how collaborative activities and their coordination can be supported by means of computer systems [24] . A common way of organizing the CSCW is to consider the work context that needs to be dealt with along two dimensions: space and time. This is the so-called CSCW matrix which identifies 4 distinct areas:

- *Same time and same place* : meeting rooms, displays and other face to face tools.
- *Different time and same place* : office rooms, large public displays or post its.
- *Same time and different place* : instant messaging [25] , video conference software, etc.
- *Different time and different place* : email, blogs, forums, and other communication/coordination tools.

As a result CSCW studies several software tools like Collaborative editors [26] and the concurrent work and use of resources by humans [27] that are significant for our current study. For more information a Handbook about CSCW, including the top cited 47 papers regarding it, can be found in [28] .

4. Scientific Artifact Platforms and Semantic Web for Research

node

Semantic Web and Social Networking Services promote the evolution of managing research resources. Nowadays, more and more online portals and software platforms provide features combining these two techniques, i.e. research social networking management with Semantic technologies. HypER [32] , Papyres [29] , Cyclades [33] ,and Mendeley [7] are the most famous web applications for scientific publications' sharing and collaborating by Web 2.0 at both data and metadata levels, while Galaxy Zoo [34] , Cohere [35] , and CORAA [36] are more focused on some certain specific procedures for scientific artifact discovering and disseminations, like indexing and search.

Besides these, there are several predefined modularities and discourse representation models (like [50]) for scientific publications and research communities which constitute the foundations of semantic annotation(typing), scientific artifact modularizing analysis(patterning) and argumentation representation. The widely used ones are as follows.

ABCDE Format

node

The ABCDE Format [31] is proposed by Anita de Waard et al., which provides an open standard and widely reusable format for creating rich semantic structures for the articles during writing. The ABCDE stands for Annotation, Background, Contribution, Discussion, and Entities respectively. Using this format, people can easily mark papers semantically, especially in the LaTex editing environment.

ScholOnto

node

The Scholarly Ontologies Project [37] led by Simon Buckingham Shum et al. in Open University aims at building and deploying a prototype infrastructure for making scholarly claims about the significance of research documents. 'Claims' are made by building connections between ideas. The connections are grounded in a discourse/argumentation ontology, which facilitates providing services for navigating, visualizing and analysing the network as it grows ([32] , [33] and [34]). They also implemented a series of software such as ClaiMaker [38] , ClaimFinder [39] , ClaimBlogger [40] and so on.

SWRC

node

The SWRC [41] (Semantic Web for Research Communities) project specifies an ontology for research communities, which describes several entities related to research community like persons, organizations, publications and their relationships [35] . It is widely used in a number of applications and projects such as AIFB porta [42] , Bibster [43] and the SemIPort project [44] . It aims at facilitating scientific resources' distribution, maintenance, interlinking and reuse.

Bibliographic Ontology [45] specifies sets of concepts and attributes for describing citations and bibliographic references (i.e. papers, books, etc) on the Semantic Web. It could be used as a bibliography ontology, a document classification ontology, or a common ontology for describing any general document. It is also compatible with many other existing document description metadata formats, like Dublin Core and so on. Zotero [46] is one of the most famous applications of this ontology.

SALT

node

SALT [47] (Semantically Annotated LaTeX) is developed by Digital Enterprise Research Institute (DERI) Galway. It provides a semantic authoring framework which aims at enriching scientific publications with semantic annotations, and could be used both during authoring and post-publication. It consists of three ontologies, i.e. Document Ontology, Rhetorical Ontology, and Annotation Ontology, which deal with annotating linear structure, rhetorical structure, and metadata of the document respectively [36] . The annotation ontology is also an extension and implementation of ABCDE format.

Part II. Knowledge and Artifact Representation

node

1. Scientific Knowledge Object Specification

node

A Knowledge Artifact is an object created as a result of an activity which encodes knowledge, which is the understanding or awareness gained beyond data. On the other hand, Complex Artifacts are those that are composed of several, simpler artifacts that have been made into a single coherent unit. Examples of these Complex Knowledge Artifacts include scientific papers, books and journals, among others.

This chapter defines and details the Scientific Knowledge Objects (abbreviated as SKOs) introduced in the LiquidPub vision document [37] . These SKOs will become the unit and our particular representation of the simple and complex knowledge artifacts in the context of the LiquidPub project.

In the light of all the known problems and limitations, explored in [4] , it is natural to wonder what would make an ideal scientific artifact format? One that is able to cope with the demands from creators, reviewers, publishers and consumers of scientific content while also enabling new interactions between these groups. More specifically, some of the requirements for this 'ideal' artifact format would be the following:

- *It is Composable and Complex* : allowing the definition of new artifacts from simpler ones, aggregating several types of data (e.g texts, spreadsheets, images, videos, etc) to facilitate the production of new knowledge or simply different executions or presentations of already existing knowledge.
- *It allows continuous evolution* : allows continuous creation and distribution of evolutions of the scientific artifact, while also supporting the current model of discrete releases at key points in time.
- *It facilitates collaborative work* : the previous properties like evolvability and composability, along with Internet-enabled interaction, greatly facilitate the creation of SKOs as collaborative effort between different actors.
- *It introduces improved models for certification and credit attribution* : provides easy-to-understand certification and credit attribution models with improved accuracy and that can be adopted gradually.
- *It is easy to find and classify* : making the large volume of information easier to filter and sort, by the use of metadata along with smart search algorithms, can be used to fight off the information overload.
- *It has a reduced work overhead* : provided suitable editor environments are created, it introduces no or very little additional work for the human actors.

The SKO-based representation system is proposed as an evolution and extension of the currently used scientific artifacts. Artifacts represented as SKOs separate in layers the different types of information that are normally or tacitly present in current scientific artifacts. Fig. 1 shows a diagram of the SKO layers.

node

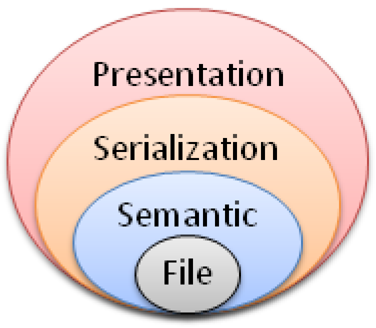


Figure 1: The different layers of the SKO specification.

node

This multi-layered approach is mainly aimed at enabling and facilitating the composition, reuse and collaborative creation of scientific artifacts. Furthermore, it provides the base for related works on improving the evolution, credit attribution, and search/navigation of these artifacts.

1. The File Layer

node

The file layer is the first and the most basic layer of the approach. Furthermore, it is also the approach's foundational layer and the main connection to well-established and commonly-used content and standards. This layer points to the actual content or data from the artifact. As such, its description starts with the explanation of a widely used concept.

URL Definition

node

URL stands for Uniform Resource Locator and provides the location or address of a particular file. By using this address it is possible to reference, use and even edit the file that it points to. Furthermore, by using HTML, multiple files (each identified by its URL) may be joined and displayed as a single page. An example of this is shown on Fig. 2 .

node

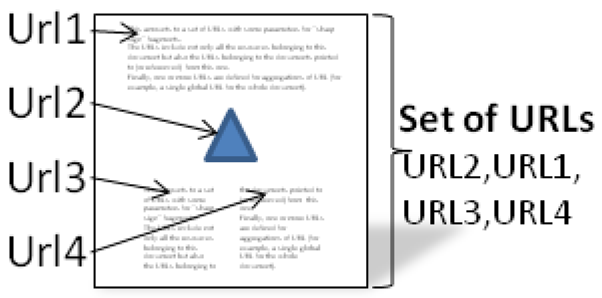


Figure 2: URLs involved in displaying a composite page.

node

Much like the citation system used in scientific publications, the URLs can also be used to reference other files or pages that, while not being included or displayed, are relevant or related to the current artifact. The URL will be used as the core component of the file layer objects, how these are defined and how they map to file-layer components will be explained in the next subsection.

File Nodes

File nodes are the approach's internal representation of already existing files. As shown in Fig. 3 , a file node is simply defined by its URL.

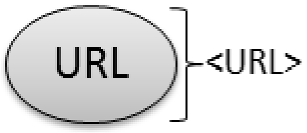
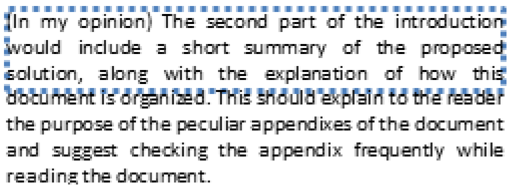


Figure 3: File node definition.

The file node can be considered as a "pointer" to the actual content that is found by using the URL. Thus, every resource that has an assigned URL, can be converted into a file node. For example, we could define a file node for the GNU GLP version 3, simply by having the URL " http://www.gnu.org/licenses/gpl-3.0.odt ", which points to that file.

File Fragment Definition

While file nodes refer to the full content pointed by the URL, there may be cases (like the ones from Fig. 4) where there is the need of referencing or rendering only a subset of that content.



(In my opinion) The second part of the introduction would include a short summary of the proposed solution, along with the explanation of how this document is organized. This should explain to the reader the purpose of the peculiar appendixes of the document and suggest checking the appendix frequently while reading the document.

Figure 4: Text fragment example, the dashed lines show the selected part.

In the same way as file nodes, file fragments can be considered as "pointers" to existing resources. Nevertheless, the file fragments do not point to the whole resource but to a sub-part of it. HTML solves this issue by using the '#' (number) sign as the fragment delimiter. For example, in " http://en.wikipedia.org/wiki/Fragment_identifier#Examples " the word "Examples" would be used as the fragment identifier (which is normally interpreted as a scrolling directive for web browsers). Note that this sort of fragment delimitation only defines the start of the fragment. However, in order to implement a finer-grained delimitation (and in accordance with other works on the subject like [38] we will also define the end of the fragment. Fig. 5 shows the fragment definition.

node

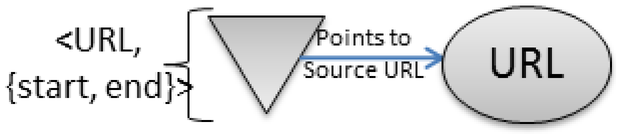


Figure 5: File fragment definition.

node

The *URL* element is used to determine the object to which the fragment refers, while *{start,end}* element is a non-empty set of start/end pointers within that same object.

2. The Semantic Layer

node

The semantic [48] layer is the second layer of the approach. Directly situated on top of the file layer, the semantic layer adds attributes and relations to the URLs, which are used to specify the specific context and concepts to which they refer. Metadata-based components from documents like the abstract, introduction and keywords can be represented or linked by using this layer.

SURL Definition

node

This approach introduces the term SURL or Semantic URL, as a means to refer to the location where the semantic metadata of an object is being stored. As such, the SURL of a given object would point to the semantic metadata (i.e. the attributes) of that object, in the same way that the URL from that same object would point to its data or content. In practice this convention allows the user to access and refer to the information on each of the layers of the object.

Fig. 6 exemplifies the uses of some of these objects to capture the semantic information related to a document.

node

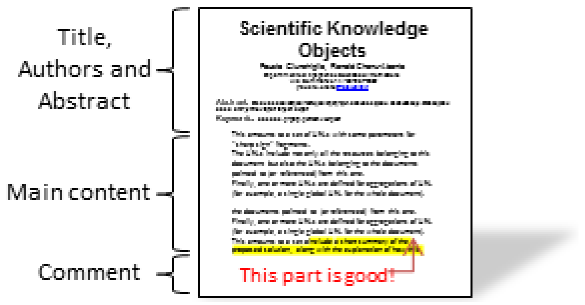


Figure 6: Example of semantic information within the first page of a document.

node

Notice that Fig. 6 is a visual representation of a first page of a document, with clearly delimited title, authors, main content and an added content referring to the highlighted part. All this information can be represented as shown in Fig. 7.

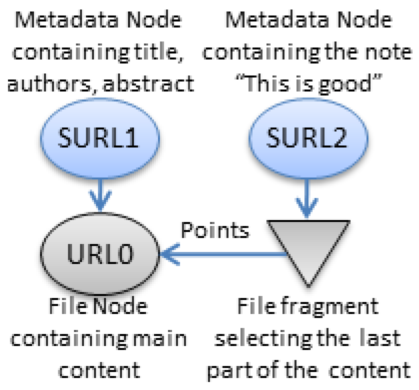

Metadata Node containing title, authors, abstract
Metadata Node containing the note "This is good"
SURL1
SURL2
Points
URL0
File Node containing main content
File fragment selecting the last part of the content

Figure 7: Representation of the semantic metadata of the first page example.

In Fig. 7 , the main content from Fig. 6 is represented with a file node (URL0) and a file fragment within URL0 corresponding to the last few sentences marked in yellow at the end of the document. Furthermore; a semantic object (SURL1) is defined containing the title, authors, abstract that correspond to the file node; while another semantic object (SURL2) containing the comment "this is good" is created referring to the previously mentioned file fragment.

Semantic-layer SKOnodes

Semantic SKOnodes are the approach's main way to enrich a file node with semantics that apply specifically and exclusively to it. Semantic SKOnodes are attached directly on top file nodes from the previous layer, and their definition is given in Fig. 8 .

In Fig. 8*SURL* and *URL* define the link between the file layer and the semantic layer. The possibly empty *{Attributes}* set is, on the other hand, used to define basic semantic information that is found on most of the SKOnodes.

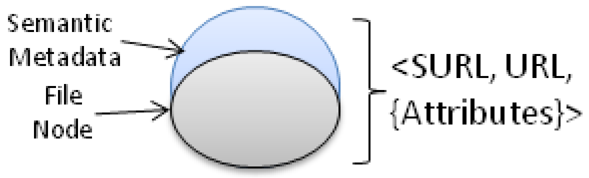


Figure 8: Semantic SKOnode definition.

An example of a SKOnode, would be the result of the objects with SURL1 and URL0 from Fig. 7 . More generally, any combination of data and semantics that refers exclusively and specifically to that data can be represented as a SKOnode.

By using semantic-layer SKOnodes it is then possible to enrich the objects from the file layer with attributes; however, the true advantages of doing so will be revealed when the last two objects from the semantic layer are explained.

Semantic Annotations

Semantic SKOnodes are used to capture fairly-common and basic attributes for specific file nodes. However, there is also the need to capture semantics that exist between objects (e.g. object A is better than object B) or that apply to more than one object at the same time (e.g. I like objects A and B). Furthermore, the need for more unique and specialized attributes is also necessary to capture less common concepts. Aemantic annotations, as defined in Fig. 9 , are the approach's response to offer the customization beyond the definition of standard attributes from SKOnodes.

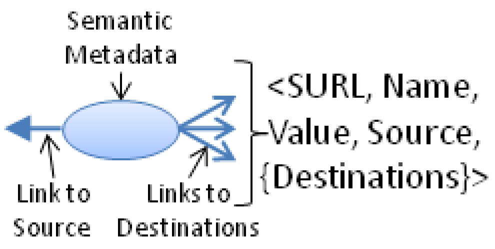


Figure 9: Semantic annotation definition.

The *SURL* element from Fig. 9 points to the current annotation, while the *Name* and *Value* elements are respectively used to identify the annotation and assign it a value. The *Source* attribute is used to store the address of the object to which the annotation mainly refers to and the possible empty *{Destinations}* set specifies the address of the secondary objects that also are affected by the annotation (examples of this below). Additional rules may be attached to each specific type of semantic annotations (e.g. the "Is related" annotation must have at least one destination). However for simplicity's sake this will not be elaborated in this document and left for the actual implementations to define. Finally, also note that neither *Source* nor *Destinations* define any restriction about the objects they refer to. As such, they may contain URLs, SURLs or the other address types referring to other objects. This enables annotations to enrich with semantic almost any aspect of an object, for example:

- *Content* : to leave a comment on the content of a document, create a semantic annotation with "Name"="Comment" targeting its file node and containing the actual comment in its "Value" attribute.
- *Semantic* : to complain about having a work being unfairly related to a given author, create a semantic annotation targeting the disputed semantic relation with the "Name"="Complaint" and containing the actual complaint in text in its "Value" attribute.
- *Style* : to suggest some visual changes to a specific paper, point to the presentation-layer object and leave a semantic annotation with the presentation suggestions.

Semantic-layer SKOs

With semantic SKOnodes covering the individual and most common attributes and semantic annotations covering the other cases, it would seem that no other object would be needed to capture information. However, as shown in Fig. 10 , there is a (quite common) case in which the semantic to capture emerge from the aggregation of objects.

node

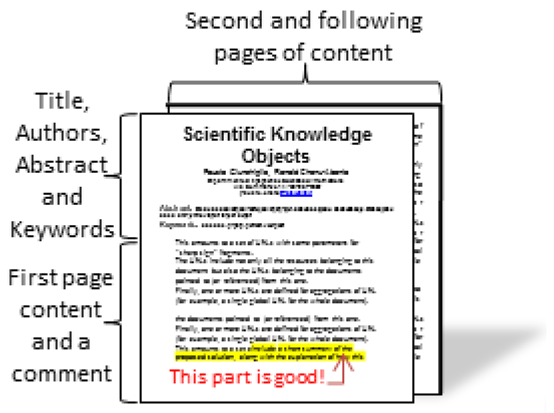

Second and following pages of content
Title, Authors, Abstract and Keywords
First page content and a comment
Scientific Knowledge Objects
This part is good!

Figure 10: Example of semantic information within the full document.

node

Similarly to the previous Fig. 6 , Fig. 10 contains a visual representation but, this time, of a full document. Fig. 11 shows a representation of this full document.

node

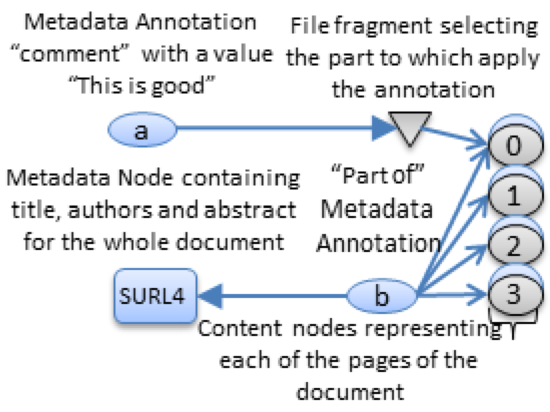

Metadata Annotation "comment" with a value "This is good"
File fragment selecting the part to which apply the annotation
a
0
Metadata Node containing title, authors and abstract for the whole document
"Part of" Metadata Annotation
1
2
SURL4
b
3
Content nodes representing each of the pages of the document

Figure 11: Representation of the semantic information of the full document example.

node

The SKOnodes: SKOnode 0 (URL0, SURL0), SKOnode 1 (URL1, SURL1), SKOnode 2 (URL2, SURL2) and SKOnode 3 (URL3, SURL3) were added in Fig. 11 to represent both the content and the semantics of all the pages of the document [49] . Furthermore, notice that a new semantic object (SURL4) containing the title, authors and abstract of the document was added along with a "part of" semantic annotation b joining it to the page SKOnodes. The reason for this is that the title, authors and abstract of the document, apply globally to all the pages of the document (represented by SKOnodes). It is key, to notice that these attributes do not apply individually to each of the pages (each page may have its own lower level title); they apply only to the whole aggregated from the pages. Thus, more than the sum of their parts, the semantic SKOs (defined at Fig. 12), capture the gestaltic (i.e. that emerge from the combination of the parts) semantics from other semantic objects.

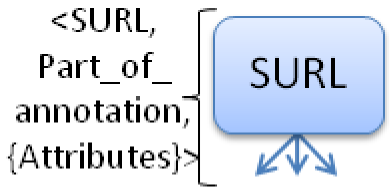


Figure 12: Semantic SKO definition.

The *SURL* from [Fig. 12](#) points to the current SKO and *Part_of_annotation* is a SURL to the semantic annotation entity representing the "part of" relation, which connects the SKO with its immediate components (in the graphical representation of SKOs, the annotation is not drawn separately for simplicity's sake). As in the case of semantic SKOnodes, the possibly empty, *{Attributes}* set is used to define basic semantic information that is found on most of the SKOs. Consider [Fig. 13](#) as an example of semantic SKOs.

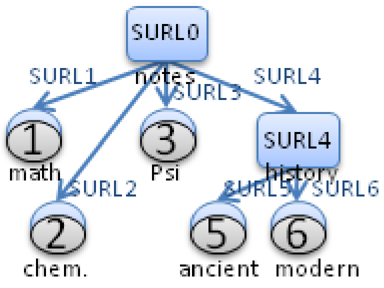


Figure 13: Semantic tree example.

[Fig. 13](#) shows a semantic tree in which two semantic SKOs and five semantic SKOnodes participate. These structures can be used to represent a categorization hierarchy as shown there or also the semantic organization of a complex document as demonstrated in [Fig. 11](#) .

Finally, while it is true that the aggregation of semantic elements could have been represented simply by using the "part of" semantic annotation alone, the semantic SKO was introduced to convey the idea that the aggregation itself is considered to be a new entity that emerged from the aggregation.

3. The Serialization Layer

On [Fig. 11](#) and [Fig. 13](#) from the previous section we have seen how the content and semantics of a complex object and each of its parts could be represented as a semantic tree. However, if there is the need of creating an artifact (e.g. a document) that contains the information in the semantic tree, it is not immediately clear how to order the data and semantics contained in it in a meaningful manner. Moreover, depending on the nature of the document that we want to create, some of the content or semantics stored in the semantic tree may not be needed (e.g. a picture used in the lecture presentation may not want to be repeated in the paper version of that same work), so we would also need a way to filter out some content and/or semantics. As the layer between the semantic layer and the presentation layer, the main objective of the serialization Layer is to organize the content and the semantics from the previous layers into a more human-friendly sequential list. This sequential list can be then converted into document (or other artifact) that effectively becomes a new whole or unit aggregated from the original parts. For example, document sections like the table of contents and bibliography (which are normally constructed semi-automatically from the full document by common word-processing tools) are introduced in this layer. By varying the information in this layer it is possible to create artifacts (e.g. documents, slides, or blogs) from the same basic semantic layer objects. These are called, different executions of the same SKO. While it is possible for executions of the same SKO to present slightly different data and knowledge, they all come from the same semantic base so they would be approximately in the same domain of information and knowledge.

A key factor to consider is that all the serialization-layer objects are presentation-neutral. That is, they select which content or semantics will be shown. On the other hand, how this information will be actually presented (e.g. fonts, colors, what format of reference to use, etc.) is left for the presentation layer. In Fig. 14 , to exemplify the use of serialization metadata, we will assume that each paragraph is represented by a different SKOnode.

LURL definition

node

Similarly to the previously defined SURL, the approach introduces the term LURL or Serialization URL as a means to refer to the location where the serialization information of an object is being stored. For the purposes of this work, LURLs will be represented as regular URLs that point to the serialization-layer information.

node

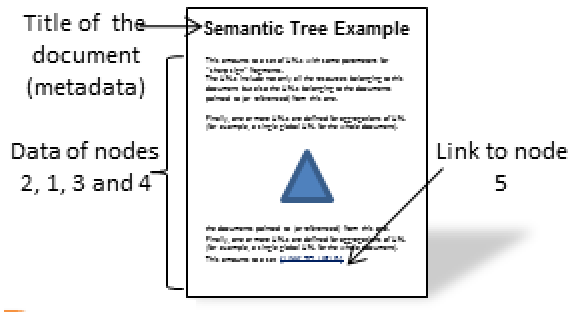


Figure 14: Example of the serialization information within the first page of a document.

node

Now, in Fig. 15 , we see an example of the representation of the serialization information from Fig. 14 .

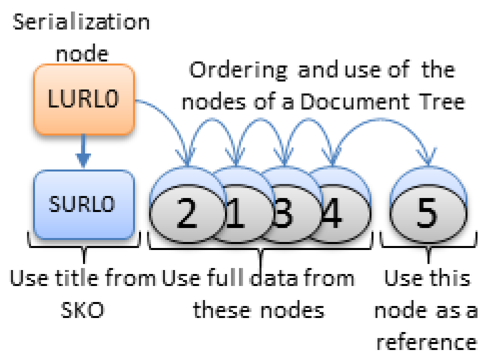

Serialization node
LURL0
SURL0
Ordering and use of the nodes of a Document Tree
2
1
3
4
5
Use title from SKO
Use full data from these nodes
Use this node as a reference

Figure 15: Representation of the serialization information within the first page of a document.

Fig. 15 shows that the documents can be represented by a SKO containing the top level semantics (title, for example), five SKOnodes (1,2,3 and 4) containing the text and images of the page and an additional SKOnode (5) that is referenced to but without actually displaying its content. Note however, that the semantic-layer SKOs and SKOnodes do not, contain neither information about what SKOnode is actually showing in the page (e.g. full content, title, references), nor the actual order in which the different SKOnodes appear. As shown in Fig. 10 , this information is contained by a serialization layer object (LURL0).

Serialization-layer SKOnodes

Serialization SKOnodes are used to define serialization information about a specific semantic-layer SKO or SKOnode. Thus, as shown in Fig. 16 , serialization SKOnodes are defined directly above semantic SKOs or SKOnodes from the previous layer.

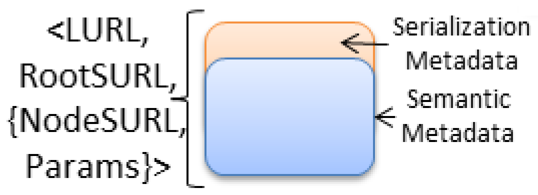


Figure 16: Serialization SKOnode definition.

The first two elements from Fig. 16 (LURL and RootLURL) are used to define the link between the semantic SKOnode or SKO from the semantic layer and the serialization SKOnode from the serialization layer. The third element is an ordered list of pairs (NodeSURL and Params), that defines which semantic objects are part of the serialization and how. The current supported values that the element Params can take are:

- *full* : includes the full content of the pointed SURL in the serialization.
- *abridged* : includes in the serialization a previously defined (via annotations, that would mark the key ideas on the text) abridged version of the pointed SURL. For example, this command would be used to include the abstract or key ideas of the document in its serialization.

- *title* : includes in the serialization the defined (in attributes) title of the pointed SURL.
- *author* : includes in the serialization the defined (through relations) authors of the pointed SURL. For example, this command would be used to include the authors of the document in its serialization.
- *reference* : with this parameter the serialization would only include a reference of the pointed SURL.

Consider, Fig. 17 as a concrete example of serialization SKOnodes, where different results are obtained from the same semantic-layer object (SURL0) by changing the serialization-layer information from the serialization SKOnodes (LURL1, LURL2, LURL3).

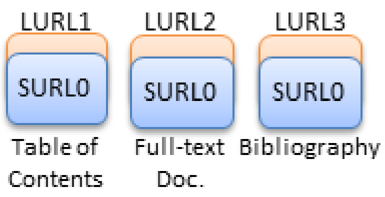


Figure 17: Example of different serialization SKOnodes.

Serialization-layer Fragment Definition

Previously to the serialization-layer we could define semantic trees (like the one in Fig. 13) to convey that multiple nodes and their aggregations were related by semantics. However, it was at the serialization-layer where we specified exactly how these subparts actually came together to form a new whole.

While serialization-layer SKOnodes refer to the full content aggregation, just like in the case of file fragments, there may be cases where there is the need of referencing only a subset of that aggregation. It is for such situations that we introduce the serialization fragment structure, as defined on Fig. 18 .

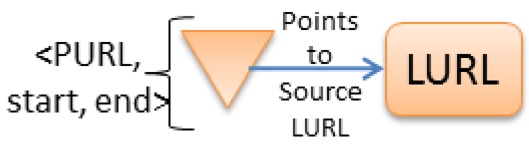


Figure 18: Serialization fragment definition.

There are a lot of similarities to be found between the serialization-layer fragments defined at Fig. 18 and the file-layer fragments defined at Fig. . These similarities stem from the fact that the objective of both fragments is to define a continuous subpart of a unitary object. In the case of the serialization fragment however, only a single start and end are defined (whereas file fragments may have many). The reason for this is that the serialization fragments operate over elements that are always linear (as they have been serialized), so only one pair of start and end arguments is needed to specify the selection of these fragments. As a concrete example of the use of serialization fragments, suppose that back at the example from Fig. 15 a concerned reviewer would like to leave an annotation that applies to the contents of node 2 and node 1. This is shown in more detail in Fig. 19 .

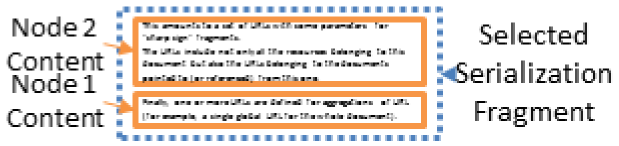


Figure 19: Serialization fragment example.

Once selected, the reviewer is able to use the fragment to leave a comment related to the serialization of the document (for example, containing something along the lines of "these two paragraphs do not form a logical progression").

Serialization-layer SKOs

Serialization-layer SKOs perform the aggregation of other serialization-layer objects. As a motivation for this, consider the example from Fig. 20 .

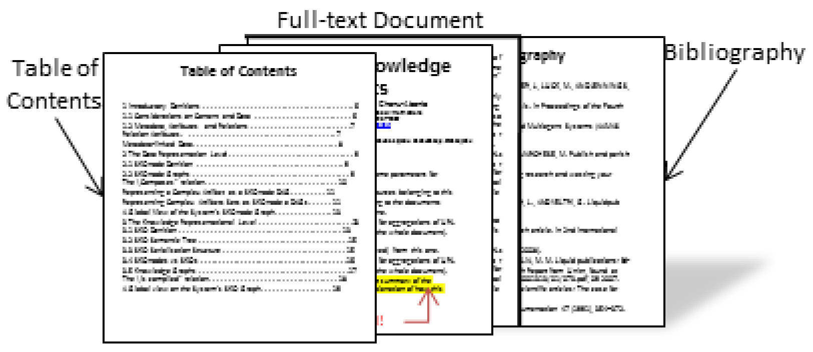


Figure 20: Example of aggregation of serialization SKOnodes.

Fig. 20 represents a full document, that has a table of contents at the beginning, then the full contents of the document and finally its bibliography. This is represented in Fig. 21 by using a global serialization object.

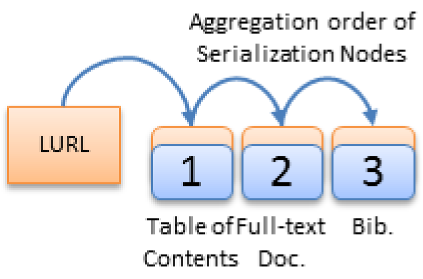


Figure 21: Representation of the aggregation of serialization SKOnodes.

The three elements from Fig. 21 (table of contents, full contents and bibliography) are represented just by three serialization SKOnodes that all have the same semantic components and only differ in their serialization. Furthermore these three serialization SKOnodes are aggregated into a single serialization structure with a single LURL. This LURL will represent the whole document and its equivalent everything in the visual representation of Fig. 20 .

The structure used in the previous object is a serialization SKO. Note that, while serialization SKOnodes are used to compose parts into new wholes (e.g. sections into a paper), serialization SKOs are used to compose wholes into new wholes (e.g. papers into journals). Serialization SKOs are defined in Fig. 22 .

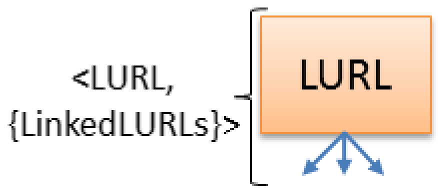


Figure 22: Serialization-layer SKO definition.

The *LURL* element from Fig. 22 defines the address for the aggregation of serialization objects, which defined by the *{LinkedLURLs}* element. The *{LinkedLURLs}* contains an ordered list of the LURLs of the aggregated serialization objects mentioned before. Note that unlike serialization SKOnodes, the serialization SKO limits itself to define which serialization objects to aggregate and does not define parameters to alter the presented content. The definition and customization of the serialization is then left to each individual SKOnode in its entirety. Consider Fig. 23 as an example.

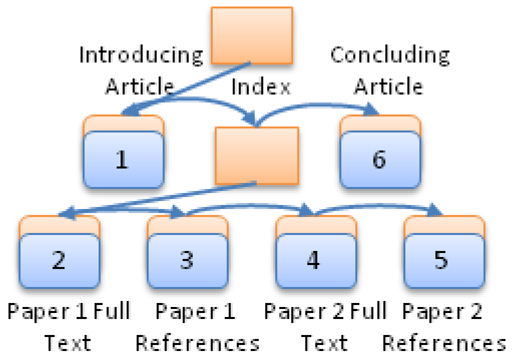


Figure 23: Serialization tree example.

The serialization tree shown in Fig. 23 is formed from the aggregation of two papers and two articles. Following the arrows we see that the order in which such document will first show the 'Introducing Article', then the full text of 'Paper 1' followed by its references, then full text of 'Paper 2' followed by its references and finally the 'Concluding Article'.

4. The Presentation Layer

This section introduces an additional layer with presentation/visualization oriented directives that enable the rendering of the previous layer's output into several presentation styles and formats. In particular, it deals with two main properties:

- *Presentation styles* : refers to colors, font types, etc. used for each type of text. For example, the headers of each section will be displayed in Times new Roman size 18, color blue. Choosing to display text in single or double column also belongs to the Presentation Layer.
- *Formats* : refers to the final output format in which all the information from the structures can be assembled into (e.g. pdf, doc, etc.). Furthermore, this layer also includes improvements in the visualization according to the display target (e.g. computer screen, mobile phone, web, etc.)

By varying the information in this layer it is possible to create different styles of the same artifact (e.g. single column, double column, text to speech, etc.) from the same basic serialization layer objects. These are called, different presentations of the same SKO. All presentations of the same SKO always contain exactly the same data and knowledge. Finally something worth noting is that, while presentation information is normally defined on top of the serialization structure (e.g. each node that represents a chapter should be rendered with the font Arial). Note that presentation information may also refer to information from other layers (e.g. render in italics all text marked as "important" at the semantic layer).

PURL Definition

Similarly to the previous layers, the term PURL or Presentation URL is introduced as a means to refer to the location where the presentation information of an object is being stored. For the purposes of this work, PURLs will be represented as regular URLs that point to files that with the extension ".pres.xml" (where the serialization information is contained).

Motivation

Fig. 24 shows, as an example, two very different presentations or styles for the same document.

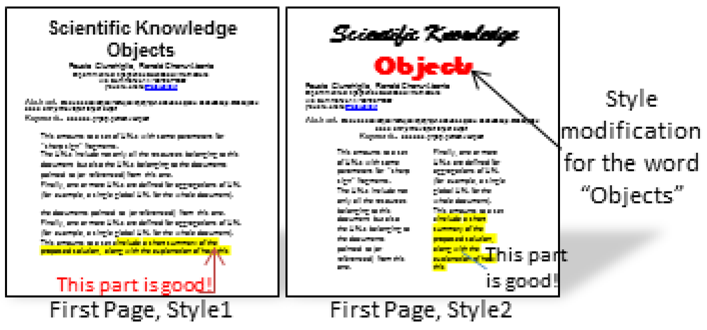
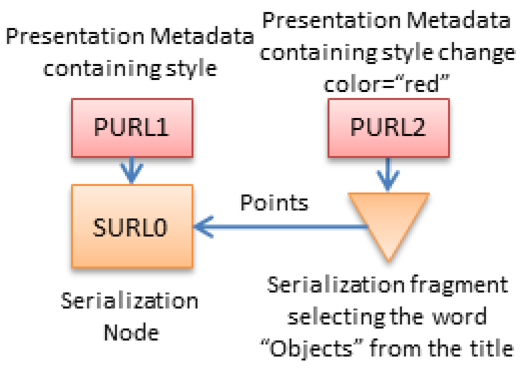


Figure 24: Example of the same document in two different styles.

Note how in the second style from Fig. 24 even the comment is presented differently and that the author wanted the word "objects" from the title to be printed in red (unlike the rest of the document). Fig. 25 shows the representation of this style's presentation information.



Presentation Metadata containing style
Presentation Metadata containing style change color="red"
PURL1
PURL2
SURL0
Points
Serialization Node
Serialization fragment selecting the word "Objects" from the title

Figure 25: Representation of the presentation information for the 'Style2' document.

In Fig. 25 , a serialization SKO (LURL0) determines the information to be displayed, while a presentation object (PURL1) contains information about the style used to present the document. Furthermore, a serialization fragment is selected to have a particular style (PURL2) that overwrites the default style defined and results in the rendering of the word "objects" in the color red.

Presentation-layer SKOnodes

Presentation SKOnodes are used to define the specific link between the serialization layer object (choosing the content) and a presentation layer object (which contains the style in which the content will be presented). The definition of presentation SKOnodes is shown in Fig. 26 .

node

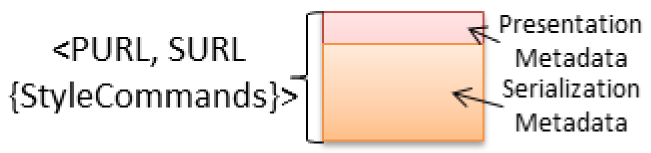


Figure 26: Presentation SKOnode definition.

node

The *PURL* and *LURL* elements from Fig. 26 define the link between the serialization and presentation objects. The possibly empty *{StyleCommands}* set is, on the other hand, used to define the presentation information specific for the current SKOnode.

A concrete example of a presentation SKOnode was given in Fig. 25 , where it was shown that the same serialization information could be presented in very different styles just by attaching different presentation-layer information.

Presentation Annotation

node

Much like the previously defined semantic annotations, the presentation annotations are used to capture specific and specialized presentation commands that may be applied to specific sections and that may even contradict the style defined in the more general presentation SKOnodes. These presentation annotations are defined in Fig. 27 .

node

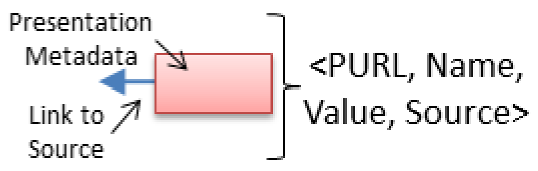


Figure 27: Presentation annotation definition.

node

The presentation annotation definition from Fig. 27 almost mirrors the semantic annotation definition from Fig. 9 . The only difference is the lack of a destination set, which is missing because unlike semantic annotations, presentation annotations always refer to a single target. An example of a presentation annotation was given in Fig. 25 , where an annotation was used to change the color of word "Objects" from black (defined in the SKOnode style) to red.

Presentation-layer SKOs

node

While presentation SKOnodes define style directives that are specific to their linked serialization object, presentation SKOs are used to define general styles that may be applied to several different documents. The definition of the presentation SKOs is done in Fig. 28.

node

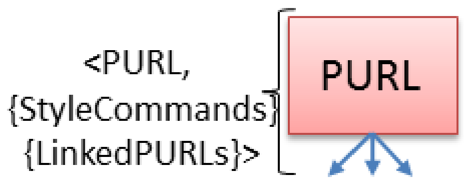


Figure 28: Presentation SKO definition.

node

The *PURL* element from Fig. 28 contains the presentation address for the SKO, the *{StyleCommands}* defines presentation directives, and the *{LinkedPURLs}* is a set of PURLs that apply those presentation directives defined. Furthermore, a single serialization may have more than one presentation-layer attached to it, as shown in Fig. 29.

node

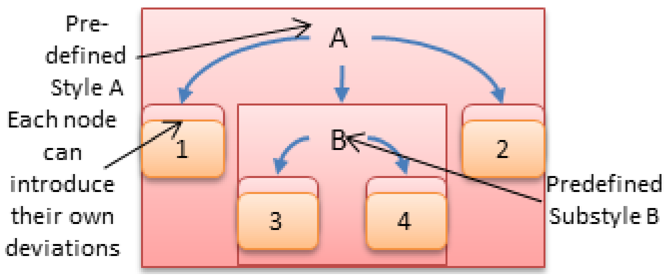


Figure 29: Presentation tree example.

node

Note that, in the example from Fig. 29 , the presentation SKO A is the most general style, which is modified individually for the nodes 1 and 2. Then, the substyle B adds its modifications to style A, while the nodes 3 and 4 introduce their own individual modifications for substyle B.

The presentation layer's main objective is to detach visual format and display consideration from the actual content and meta-information as much as possible. This would enable content creators to generate artifacts that are able to be represented in several display formats (e.g. ACM, LNCS in the case of paper representation formats) or aimed at different display devices (e.g. computer screen, printer, mobile computer screen) without much additional work.

2. Features and Services Enabled by the SKO Specification

node

There are several operations and services that can be defined on top of the SKO specification, it is through these services that the true representation power and usefulness of the previously defined structures can be fully demonstrated. Note that the discussions here discuss the general aplication of these features and services, while the concrete implementations of these features and services will be given in the application-related parts of this thesis.

1. Version Control and Branching

node

Using the previously defined semantic structures, it is possible to create tags that would be useful to identify when an artifact is a version of another artifact. As in other version control systems, the same mechanisms can be used to determine whether an artifact was merged or split from previous ones. However, when considering the SKO layered structure system, interesting options like part/whole and layered version control are also enabled.

Version Annotations

node

The first version control encoding annotation is the "is version" annotation. Fig. 30 , exemplifies a situation where the use of such annotations would be necessary to track the evolution between objects.

node

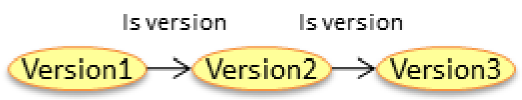


Figure 30: A simple version contro evolution example.

node

As all semantic annotations from the SKO specification, the "is version" annotation is essentially defined by the following elements:

- *Name* : in this case, this value is always set to "is version", this helps identify the semantic annotation as being in charge of version tracking.
- *A single source* : for version tracking annotations, the source points to the original or source object that was evolved into one or more additional objects. In Fig. 30 the source of the first version control operation from the left would be "Version1".
- *Multiple destinations* : the destination set contains all the objects that were versioned from the object pointed by the source property. This includes both direct versioning (like the one existing between Version1 and Version2 in Fig. 30) and indirect versioning (like the one existing between Version1 and Version3). It is not known whether this redundancy will be found practical at the system's implementation time but it is kept for the moment, as it makes easier to determine the version control history of a given objects.
- *Value* : unused for version control annotations.

By using the previously defined version annotations it would be possible to represent the example from Fig. 30 into whats found in Fig. 31 .

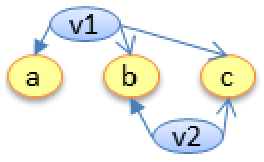


Figure 31: Representation of the previous version control example.

Fig. 31 has three nodes (a, b and c) representing the artifacts and two annotations (v1 and v2) representing the actual "is version" annotations. Note that the annotation v1 conveys that both b and c are *newer* versions of a, while the annotation v2 conveys that c is a newer version of b.

The following are some additional properties that the "is version" annotations have in their current implementation from D1.3:

- *Unique source* : this means that the only one "is version" annotation is allowed to exist for each source object. This restriction has the implication that all the tracking of the derived versions of a single object will be tracked in a single annotation (this may change however in the future due to implementation and scalability issues).
- *Fragments are allowed* : both the source and destination of a "is version" annotation may refer to fragments of objects instead of full objects.
- *Recursion not allowed* : a single object is not allowed to be a version of itself.
- *Source and destination entities* : the source and destination from a "is version" annotation must both refer to the same type of entity (e.g. if source is a SKOnode, destination must be a set of SKOnodes).

As exemplified in Fig. 32 , two additional semantic annotations are necessary to help identify branches, forks and splits in the version evolution of an object.

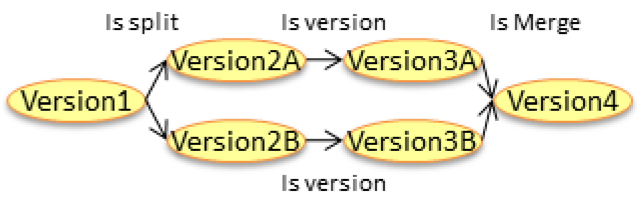


Figure 32: A simple branch/fork and merge example.

The first annotation type to be introduced is "is split", which is used to represent branches and forks. Split tracking is implemented as a semantic annotation, essentially defined by the following elements:

- *Name* : in this case, this value is always set to "is split", this helps identify the semantic annotation as being in charge of tracking conceptual fork and branches.
- *A single source* : for split tracking annotations, the source points to the original or source object that was split into one or more additional objects. The source of the split found in Fig. 32 would be the "Version1" object.

- *Multiple destinations* : the destination set contains all the objects that were created as a split from the object pointed by the source property. The destinations of the split found in Fig. 32 would be the "Version2A" and "Version2B" objects.
- *Value* : unused for "is split" annotations.

The second annotation is "is merge", which is used to represent the merge of two objects into a single one. Merge tracking is implemented as a semantic annotation, essentially defined by the following elements:

- *Name* : in this case, this value is always set to "is merge", this helps identify the semantic annotation as being in charge of tracking the conceptual merging of entities.
- *A single source* : for merge tracking annotations, the source points to the actual merged entity. The source of the merge found in Fig. 32 would be the "Version4" object.
- *Multiple destinations* : the destination set contains all the objects that were merged into the object pointed by the source property. The destinations of the merge found in Fig. 32 would be the "Version3A" and "Version3B" objects.
- *Value* : the value of "is merge" is used for storing an identifying tag or a reason for the merge. For example, the value for the merge annotation from Fig. 32 could be "reunification of the work done by group A and B".

By using the previously defined version annotations it would be possible to represent the example from Fig. 32 into what is found in Fig. 33 .

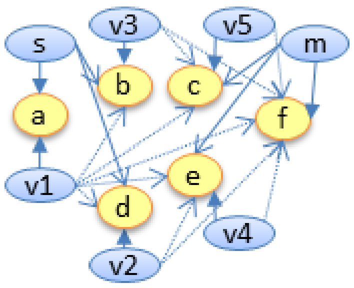


Figure 33: Representation of the previous branch/fork and merge example.

Fig. 33 has six nodes (a, b, c, d, e and f) representing the artifacts and seven annotations (v1 to v5, s and m) representing the version control relations. The list below will clarify the meaning of each of these annotations:

- *v1* : "is version" annotation that conveys that b, c, d, e and f are all newer versions of the artifact a (i.e. a is the version predecessor of them all).
- *v2* : "is version" annotation that conveys that e and f are both newer versions of the artifact d.
- *v3* : "is version" annotation that conveys that c and f are both newer versions of the artifact b.
- *v4* : "is version" annotation that conveys that f is a newer version of the artifact e.
- *v5* : "is version" annotation that conveys that f is a newer version of the artifact c.
- *s* : "is split" annotation that conveys that b and d are both splits from the artifact a.
- *m* : "is merge" annotation that conveys that f is a merging of c and e.

Similarly to the "is version" annotation both, "is split" and "is merge", have additional properties and limitations applied to them in the current D1.3 implementation that (in the interest of maintaining the generality of this chapter) will not be elaborated here. A detailed specification of this will be left to the concrete application that implements this SKO-based version control.

A final key point to note is that all the annotations introduced here do not actually capture how the artifacts are split, merged or changed. These annotation are more concerned on capturing the occurrences and reasons for this events at the conceptual level. This information will be later be put to use to enable platform features like search and navigation, among others.

Layered Version Control

node

As in the SKO the artifacts are composed of several objects in different layers, it would be possible to version and change only some layers of a given artifact without introducing any changes to the rest. For example, changing serialization-layer objects would create a different representation of the original artifact (with more or less the same concepts but different granularity/order). Like in the regular version control, the layered versions are represented by semantic annotations. Nevertheless, whereas in regular version control both the source and destination are in the same SKO layer, in the layered version control the source and destination are from different layers.

The execution relation is the main connection between the semantic-layer and the serialization-layer objects. It is mainly used to create different documents from the same general repository of content and knowledge. Fig. 34 shows a representaion of Fig. 17 as an example on how the serialization-layer objects interact with semantic-layer objects to create different types of artifacts from the same knowledge and content.

node

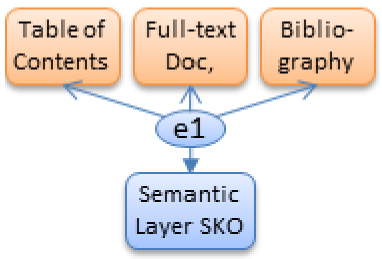


Figure 34: Example representation using the "is execution" relation.

node

As all semantic annotations from the SKO specification, the "is execution" annotation is essentially defined by the following elements:

- *Name* : in this case, this value is always set to "is execution", this helps identify the semantic annotation as being in charge of execution tracking.
- *A single source* : for execution tracking annotations, the source points to a semantic-layer object that will be executed into one or more of serialization-layer objects.
- *Multiple destinations* : the destination set points to one or more serialization-layer objects that are the execution (i.e. contain the serialization-layer information for that object) of the semantic-layer object from the source.
- *Value* : unused for is execution annotations.

2. Search and Navigation

node

This section will discuss the searching and navigation possibilities that are enabled by the SKO specification. Concrete implementations of search and navigation over SKO objects will be further specified in the concrete applications discussed on the following chapters. As such, this section will focus on detailing the general information and features from the specification related to Search and Navigation that are available to allow each application to implement for their own specific needs.

Search options

node

The following are search strategies based on each of the three layers of the SKO specification. These are not really meant to be carried out separately but to aid each other in order to offer the possibility of searching for unique aspects and allowing the return of the most adequate results to the user's query.

- *Content-based* : content-based search is mainly based on the structures from the file layer. This represents the, normally computing expensive, conventional full-text search. As such, this should be only used after narrowing down the search space with the previous methods. Note that for the moment, this type of search is only restricted to text-based artifacts.
- *Semantic-based* : based on structures from the Semantic Layer (SKOs, SKOnodes and annotations). At its most basic implementation, this search can be used to query for specific values in semantic and relations that are common between the searched entities. Some examples of this include keyword search (using the keywords encoded as semantic) or related document search (which queries the existing relations between the items such as comments, citations, older/newer versions). Once a good body of material is in the system and enriched with semantic, more advanced versions of this concept-based search can be used by querying discourse representing relations between entities or even the opinions and comments encoded as semantic.
- *Pattern-based* : centered around serialization-layer objects that represent common patterns of complex objects. Searches of this type would involve searching for documents that follow specific structural or discourse patterns. For example, pattern-based search query would be "documents that follow a Deductive pattern" or "documents that follow the Introduction-Body-Conclusion structure".

These types of queries used together lead to the creation of more complete and realistic query examples. For example "documents from Fausto Giunchiglia that follow Deductive Reasoning".

Navigation options

node

The SKO specification, uses the semantic-layer annotation to keep track of the conceptual relation between objects and their evolution. These annotations can be, in turn, used to enable interesting navigation options like the following:

- *Citation-based* : with enough data bootstrapped, citation networks that include not only papers and books but also other types of documents (e.g. presentations, webpages, etc.) would emerge. Being internal, these would be straight to update and navigate.

- *Version-based* : an evolution line, tree or graph (depending on the branch/merge history) would be able to be created based on the different existing versions of the same artifact and its components. Furthermore by using the state-based SKO maturity measure (defined in the following chapter) as a filter and semantic-encoding evolutions, the evolution of knowledge along its different maturity levels would be much easier to see and navigate.
- *Component-based* : massively complex components (that have several levels of composition) will be made easier to navigate thanks to the creation of composition trees (which are not unlike the folder trees of current operating systems).
- *Execution-based* : when a single idea or base document is converted in several executions (e.g. conference paper, presentation, journal paper, one-page abstract, book), having a navigable execution-based graph would greatly help the user to find the granularity-level that he or she is interested in reading.
- *Discourse-based* : once enough semantic information is in the system, it would also be possible to create and navigate "knowledge networks" that will represents ideas and points of view as nodes and link them relations such as "supports", "is opposed to", etc.

3. Ownership and Credit Attribution

node

This section explores how the previously detailed SKO specification can be used to enable the concepts related to ownership and credit attribution.

The basic way in which the SKO specification tracks authorship is by having an "author" annotation (i.e., semantic annotation, as previously specified) applied to both SKOs and SKOnodes of each layer. However, this straight-forward authorship tracking is not sufficient to cover all the details of the options enabled by the different granularities of structures and their different layers introduced in the SKO specification. As such the following list will explore some options enabled by the SKO specification:

- *Granularity of Credit Attribution* : the SKO specification allows sub parts of a complex scientific resource may have different authors.
- *Credit Attribution for Non-authoring Roles* : roles like providing experimental data, writing, editing and typesetting of the document are traditionally all "rolled-up" into the author role. The different layers of the SKO structure can also be particularly aimed at some of these roles. For example, contributors that work on File and Semantic layer entities from create content (e.g. datasets, text, figures) and semantic material related to it; while the contributors that work on the serialization layer normally take editing roles (i.e they select and order already existing content without actually changing the content themselves).
- *Add new roles or qualification of the interaction* : new roles that do not directly correspond to the layers of the SKO specification or even with the document creation (i.e. they happen before or after the creation process of the document) can also be captured through the use of annotations. Furthermore, other annotations may be introduced for registering how mayor a contribution to the overall scientific resource.

4. Licensing and Copyright

node

SKOs are also able to represent original works. As such, copyright laws and licensing regulations are also applicable to them. Much in the same way as authorship the basic copyright and licensing information is kept at each of the SKOs' and SKOnodes' semantic.

Conventionally, documents (like papers and books) are treated with "blanket licenses" that cover the whole document and all the content (e.g. figures, text, tables) that is part of it. Nevertheless, as it is in the case of credit attribution, the SKO specification allows for licensing and copyright information to be defined at lower granularities. This would be similar the way a software artifact is composed of modules and libraries that may have different (and even contradicting) licensing information defined for them. This is further complicated by the additional segmentation of contributions introduced in the SKO specification. For example with the SKO specification in place, it would be possible to define different licensing and copyright rules for the actual content of the document and another for the document pattern being used.

The SKO-based application side-step the issue of Copyright by never duplicating the actual content (e.g., pdf files for papers). As such, the original Licensing and Copyright rules of the original content are kept (for example, a paper would only be browsable in the application if the user originally had access to the material) while the bulk of all services and processes is offered based on the metadata related to this content.

3. Possible Extensions of the SKO Specification

node

This chapter details proposed features of SKO-enabled systems that, while not directly applied or validated in this thesis, still remain interesting and potentially useful.
A much more exploratory approach will the be used to discuss these features and propose them as future extensions to the SKO specification that may be considered for implementation or as a reference point for future applications and research.

1. Maturity Measure (States)

node

This section introduces three states (Gas, Liquid and Solid) an additional dimension extending the multi-layered structural approach previously presented. In order to make easier to understand some of the more abstract and complicated properties from scientific artifacts (e.g certification, persistence), this section introduces an easy-to-understand metaphor based on the most well-known physical states of matter. As in the physical world, an object may have very different properties depending whether it is in one of the three discrete states (Gas, Liquid and Solid) introduced for scientific artifacts (as shown in Fig. 35).

node

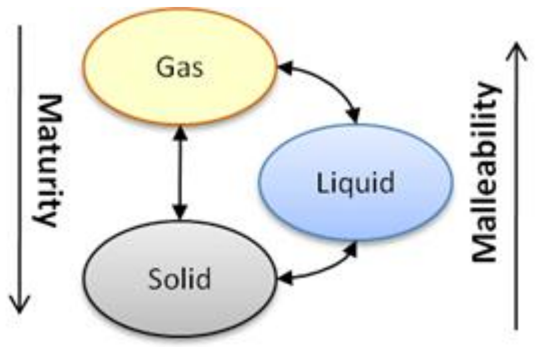


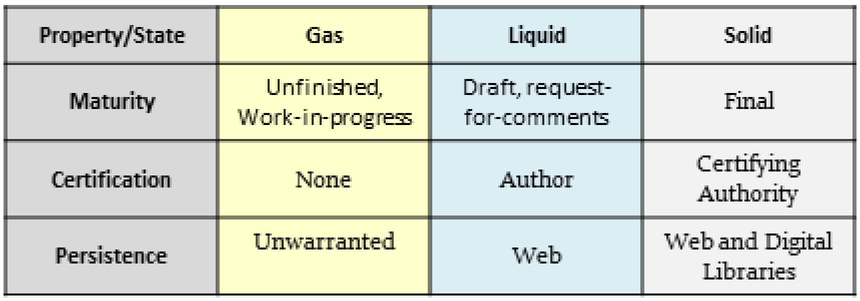
Figure 35: SKO states diagram.

node

The key properties that define each of these states are contained in Tbl. 1 shows a diagram of the SKO layers.

node

Table 1: Main properties of the three SKO states.



Property/State	Gas	Liquid	Solid
Maturity	Unfinished, Work-in-progress	Draft, request-for-comments	Final
Certification	None	Author	Certifying Authority
Persistence	Unwarranted	Web	Web and Digital Libraries

node

In more detail the main three properties abstracted away in this maturity measure are:

- *Maturity* : despite being a basically subjective property, maturity is the most representative and intuitive of all of state properties. Gas objects are normally used for highly fluctuating work-in-progress and deemed with not enough maturity to be considered serious. On the other hand, liquid objects are considered to have all their basic knowledge or science in place but still undergoing some adjustments, while solid objects are considered mature enough for being candidates for extending human knowledge or science.
- *Certification* : refers to the person or entity that takes responsibility and, eventually, credit for the content of the object. As shown in the previous table, it is the author that certifies liquid objects; which means that he assumes responsibility for the content in it. For solid objects, both the author and a certifying authority (e.g. a publisher a board of reviewers, etc.) assume responsibility for the artifact.
- *Persistence* : refers to the method used to warrant the availability of the objects over a period of time. This is important, for example, for citation-based processes; which need their objects to be relatively persistent to perform adequately. In a platform warranted persistence (liquid state) the SKO platform itself warrants that all liquid objects will remain available. On the other hand, in a "platform and digital libraries" warranted persistence, the object may be distributed and duplicated in digital libraries to further improve its availability.

The following subsections will further clarify and exemplify the three introduced states.

The Gas State

node

By creating objects in the gas state the authors themselves admit that the content of such objects is a work-in-progress, immature or untested. As even its authors do not certify the content of the objects in this state, it is considered having the lowest maturity and is not generally deemed important enough to ensure its persistence. This means that no reliable reputation (credit/blame) or citations can be derived from this type of objects. Nevertheless, these properties also allow gas-state objects to be the most malleable of all states; making them the default starting point for new ideas. The following list explores additional details about the gas state:

- *Main Purpose* : objects in the gas state are mainly used as the starting point for new ideas or for introducing changes to already established work. Frequent and unordered changes, are expected in this state.
- *Evolution* : since no persistence is warranted for gas state objects, they are in general the least constrained objects in the system. As such it is common to allow the deletion and modification of objects in the gas state, despite the fact that it may be referenced or used by other objects.
- *Target audience* :gas objects are normally aimed to a very reduced number of trusted collaborators. This is caused by the frequent and loosely tracked modifications, along with the preliminary and evolving nature of ideas contained in this state.

A current example of what we call a gas state object would be a work-in-progress document stored in on local media and being worked on by the author exclusively or by a small group of collaborators over a LAN. The software equivalent of this state would be a development version of an application.

To start a more concrete example, consider the fairly common case of a scientific author writing a paper for submitting it to a conference. At the beginning author would create a first version of the paper that he wants to eventually submit (version 0.1). The author would then keep working on this gas state document (through versions 0.2, 0.3 and 0.4) until he finally reaches a point (version 0.5) where he feels that the theory and ideas of the paper are relatively stable and conveyed in a good enough manner. The author at this point would be interested in receiving feedback of his current progress, he is however concerned about protecting his ideas and giving this work a little more of persistence (which would enable interested parties to reference or contribute this work). This is when the author would need the liquid state, introduced in the next subsection.

The Liquid State

node

Authors creating objects in the Liquid State generally acknowledge their work as not being final but still they deem it stable enough for partial dissemination. To protect the correct credit attribution of the ideas and content that they contain, liquid-state objects demand personal certification from its authors (authors take the responsibility and credit of the liquid-state content they have produced). Finally, the persistence of liquid-objects is enhanced to allow their possible (according to the access-control configurations set by the creators) citation and reuse.

The following list explores additional details about the liquid state:

- *Main Purpose* : much like a "beta" or "candidate" release in the software world, the liquid-state objects constitutes an important chance for the creators of getting important feedback that will be used to produce a final version. Another important purpose is updating the members of your community about the progress you have made on a given subject. So in that sense, liquid objects could also be considered as temporal increase in the human knowledge until a solid version is released.
- *Evolution* : persistence of the liquid-state objects is now warranted by the platform they have been published on and each liquid release will probably be kept for reference and archival even though a newer one may have been released (unlike in gas-objects in which their low persistence requirement allowed them to be freely deleted and overwritten). Furthermore, because of their maturity standards, the author is expected to release liquid-state versions of their objects in a somewhat less frequent manner (unlike internal gas-object that may get daily or hourly releases). Finally, providing the equivalent of "patch notes", or a list of the

main changes done since the release of the last liquid version of an object could be either mandatory or strongly suggested (depending the platform's requirements).

- *Target audience* : the previous characteristics enable to disseminate the object within a reference community in order to obtain feedback. Note also that it is the a finalized liquid-state object what will be submitted to a certifying authority/reviewers when the authors want to obtain the permission to release a solid state version.

Currently there are no widely-used examples of liquid-like scientific objects. However, software equivalent would be a Beta testing release of an application. This is why, one of the main objectives in the approach presented in this document is the introduction of the Liquid state objects, as a bridge between currently existing and widely used gas-like and solid-like objects.

Continuing the example of the author evolving a paper for submission to a conference, the author has just released a liquid version of his latest document (version 0.5). This liquid document quickly gets some feedback from the author's friend, as shown in Fig. 36[51] .

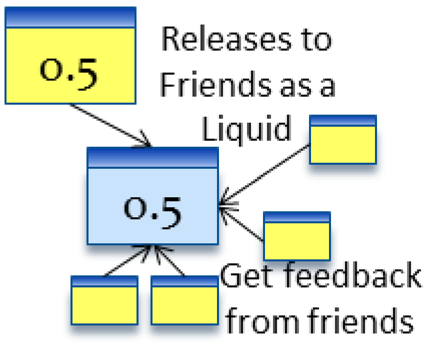


Figure 36: A liquid version is released to obtain feedback from friends.

While he/she waits for feedback the author continues to work and creates an internal (i.e. for his own use only) gas version (version 0.6). Later the author decides to include the feedback obtained into creating a new internal version (version 0.7), as shown in Fig. 37 .

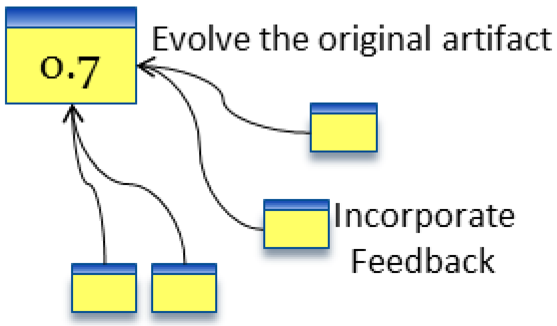


Figure 37: The feedback is used to evolve the original document in progress document.

After creating a new internal version (0.8) for addressing some finishing touches, the author is satisfied with the current iteration of his work and decides to release a liquid state document. This time (as shown in Fig. 38) he decides to submit it directly to the conference, where he gets reviews for that version.

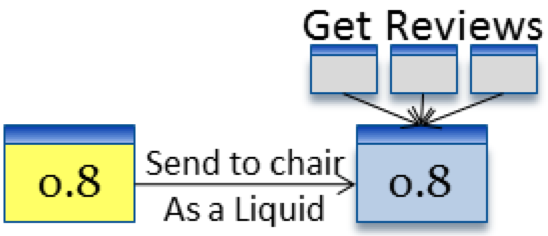


Figure 38: A liquid version is sent to the chair of a conference and gets reviewed.

The reviews that the author receives include interesting feedback and suggestions that he addresses in a new internal version (0.9), as shown in Fig. 39 .

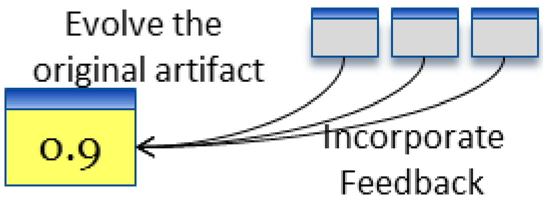


Figure 39: The review feedback is used to make correction and evolve the original document.

After the review round and the implementation of the reviewers' suggestions, the author addresses the last pending issues by releasing a final version (1.0). At this point, the author is confident that he/she has a high-quality and stable version of his work (version 1). Therefore, he/she is now interested in releasing and disseminating this work so it can be discussed and used by the general public. These are properties that belong to the solid-state objects that will be discussed in the next subsection.

The Solid State

As the state that is characterized by having the highest maturity, solid-state objects represent a milestone of "maturity" in the evolution of an artifact. To ensure this maturity and their quality, solid-state objects cannot be self-certified and are normally certified by higher-level organization (like a journal, conference or a publisher). Furthermore, solid-state objects are considered very important reference material. Therefore, their persistence is normally ensured by duplication in one or more (digital) library repositories.

- *Main Purpose* : the main purpose of solid-state objects is the increase of science or, more generally, of the human knowledge. As such, solid-state objects are the basic unit for dissemination and consumption for the general public.
- *Evolution* : as in the liquid case, state object persistence is also warranted for solid state objects, which allows every version of a solid object is remains available for reference. The

only difference here would be that solid state objects generally tend to have far less versions that liquid state objects, as the certification procedure involved to produce a solid version is normally far more complicated/expensive.

- *Target audience* : the maturity and certification properties of solid objects ensure that they are ready for general public releases and wide dissemination.

The documents released through current publication practices (e.g papers, books, journals) are current examples of solid state objects.

As the conclusion of conference submission example, the author once again submits to the conference (this time the version 1.0 of his work). There, as shown in Fig. 40 the conference organizers finally approve its work and release a solid version (1.0) of this work.

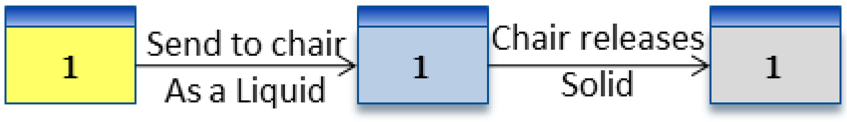


Figure 40: A new liquid version is sent to the conference chair. The chair approves and releases the certified solid version.

Based on the previous definitions it is clear that solid state artifacts (or artifacts with equivalent properties) are already widely used and distributed (e.g. published papers, hard-back books, etc.). At the same time, gas state artifacts are also ubiquitous if one also considers the storage of personal computers (e.g. work-in-progress document, unordered notes). It is however much more rare to find (especially in scientific environments) the use of artifacts with properties similar to those described for the liquid state.

As such, one of the key points of this feature is the introduction of the liquid state objects (e.g. request-for-comment and intermediate-level documents) as a way to improve the collaboration and early dissemination in the scientific process. Furthermore, by introducing a new less demanding "personal-level" certification and allowing the reviewing/feedback loops to start from that point and thus enabling early dissemination.

2. Version Control for Aggregation

The clear identification of parts and wholes for each layer that the SKO specification includes, allows the version control system to track the evolution of each artifact and its aggregated artifacts almost independently. Software artifact families are a clear example of this type of part/whole evolution, and examples of these can be found at [39] .

Changes on aggregated objects

Introducing changes on aggregating objects refers to the action of adding or modifying information from the aggregated objects (or parts) and to the consequences this may bring over the objects aggregating them (or parts).

Changes in the aggregated objects may happen in any of the four layers defined in the SKO specification, the changes happening to them would affect the aggregating objects from each of these layers in various ways. For example, if the text of a paragraph that it is used in both a paper and a book is modified this will cause the content of the paper and the book to change too. Expanding the previous example, if that same paragraph is linked to by annotation that states that it is used to prove a theory introduced in a presentation; then both the aggregating book and paper (up to certain degree) can be said to also help to prove the theory on the presentation.

Since it is clear that changes in the parts affect the wholes deeply, the following control methods could be implemented (according to the needs of the particular use case):

- *Auto-forking* : the owner of the aggregating object may choose to ask the system to automatically fork its components into a separate copies should they get modified in anyway (as exemplified on Fig. 41). By doing this the owner of the whole guarantees that the information of the wholes remains the same despite changes in the aggregated objects (but at the risk of pointing to outdated information while newer and more correct information is already available). Note that this case would be the preferred for the creation of very complex and collaborative artifacts, like most scientific documents.

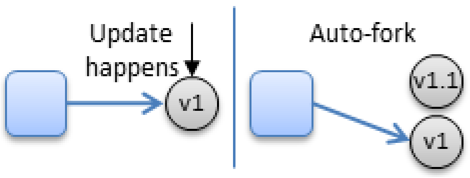


Figure 41: Auto-fork example: as soon as a modification is introduced for the pointed component, a fork is done by the system to preserve the information being aggregated.

- *Auto-updating* : the owner of the aggregating object may choose to ask the system to always point to the latest version of its components (as exemplified on Fig. 42). By doing this the owner of the whole guarantees that the most up-to-date information will always be propagated to the whole (but at the risk of pointing to less stable or finalized information).

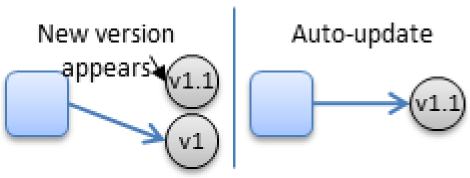


Figure 42: Auto-update example: as soon as a new version of the component appears, the system changes the "part of" annotation to point to it.

Note that these behaviors may be configured to operate in conjunction with the previously defined maturity-based states. For example, if modifications revert a component to the gas state; then it is probably better to just auto-fork to avoid introducing unstable or immature information into the aggregating object. On the other hand, if the system notices that a new available version of a component is liquid or a solid component; then it would become safer to auto-update to it. In any case, all changes and new versions of components should be always notified to the aggregating object owners. This enables the them to avail the impact of the change and to evaluate the proper course of action to take (much like a software programmer would be interested in being notified about updates to the libraries that his software uses).

Changes on aggregating objects

node

Similarly to the previous subsection, introducing changes on aggregating objects refers to the action of adding or modifying information from the aggregating objects (or wholes) and to the consequences this may bring over their objects being aggregated (or parts).

According to the previously discussed SKO specification, there is no file-layer aggregation (i.e. file-layer SKOs) defined. This means that in the SKO specification the aggregating objects never introduces new file-layer content (e.g. the aggregating object cannot add a new picture that is not present on any of the aggregated parts); it can however introduce new information in the semantic-layer (i.e semantics and relations that emerge from the aggregation), the serialization-layer (i.e. content choosing and ordering) and in the presentation-layer (i.e styles to be applied to all original content).

But even if aggregation does not add new content to each whole, changing the semantics and the relations from the aggregating wholes could have implications on the aggregated parts. For example, if a reviewer adds his comments to a paper; these may also be relevant to at least some of the components of this paper.

In general however, at least for the moment, these types of modifications are deemed more as an informative tool than a content altering one. As such, having the system send notifications to the component owners about changes in the aggregating object should be enough to regulate these interactions (much like a software library creator would like to get notified when other applications start using or modify the usage of their libraries).

Part III. Capturing the Scientific Lifecycle with SKOs

node

1. A Metadata-Based Platform for Knowledge Production and Dissemination

node

node

Scientific papers and scientific conferences are still, despite the emergence of several new dissemination technologies, the de-facto standard in which scientific knowledge is consumed and discussed. While there is no shortage of services and platforms that aid this process (e.g. scholarly search engines, websites, blogs, conference management programs), a widely accepted platform used to capture and enrich the interactions of a research community has yet to appear. As such, we aim to explore new ways for the members and interested people (persons) that create or discuss their work (papers) in research communities to interact; before, during and after conferences (events). These papers, persons and events, represented as metadata-enriched entities, are used not only to manage a large body of new and legacy papers but also for aggregating useful information and services related to the persons and events.

1. Introduction

node

The main part of the conceptual groundwork for the platform presented in chapter is based on the previously defined SKO (Scientific Knowledge Object).
As an example of similar approaches and of the general objective of this platform, the following are some of the services that have already been created to aid scientific production:

- *Search engines* : a search engine specifically tailored for scientific papers, persons and events (e.g. Google Scholar [55]).
- *Conference websites* : these sites contain announcements and concrete information about the venue and event of a conference. They are also mainly used to register (and pay) for participation (e.g. the ESWC2011 site [56])
- *Conference series websites* : these types of sites contain proceeding (i.e. collections of papers), give various announcements and offer special pages for the organizers to share information (e.g. the JCDL conference series site [57]).
- *Submission management* : used to manage the submission process of papers to a conference (e.g. Easychair [58]).
- *Social Networks* : like Facebook [27] , almost unchallenged as a recreational and commercial social network. We would also want to offer scientific-related resources and content as the very center of the social network (e.g., Academia.edu [4]).
- *Digital Library Portals* : like ACM's [6] include several of the features we are interested in (albeit sometimes in a reduced or more basic manner).

The wide variety in the approaches and services mentioned shows that many of the functionalities and services that are considered interesting for scientific research are still somewhat dispersed. It would be normal for a researcher, for example, to first search for a paper in scientific search engine, then find it in the proceedings of a conference site and finally resort to email communication to comment or discuss. Besides the minor inconvenience of using multiple sites, the main problem is that no explicit tracking is kept over the whole process or even the interactions or discussions that are taking place. A platform used to capture and enrich the interactions that happen inside the research community has yet to appear, that is able to manage a large body of new and legacy papers while also for aggregating useful information and services related to persons and events from the community.

To bridge the seemingly antagonistic, conventional approach and the web-enabled new technologies, we propose the creation of a community platform focused not only around the papers that contain the scientific knowledge or the persons that create it, but also around the events in which these persons meet to discuss this scientific knowledge. By using these three key elements, abstracted as metadata entities (i.e., a specific representation of objects in the real world) and enriched with semantic technology, we aim to accomplish the consistent aggregation of previously disjoint services; like providing access to conventional papers, provide information to aid and enable visiting conference events and, offer services that would enrich the interactions and contributions with other researchers.

In particular, we want to create new ways for the members and interested people working in the Artificial Intelligence (AI) research community to interact; before, during and after their conferences. To serve as a base to these interactions, we want not only to obtain, format and manage a body of legacy and new papers related to this community but also to aggregate several (previously dispersed in several sites) useful information about persons and events to the environment of a community platform. This platform, which was previewed during the International Joint Conference on Artificial Intelligence (IJCAI) 2011 and is still under development, would allow the members the community to discuss their work and to share their content (e.g. presentations, videos, notes and pictures).

2. Entities

node

One of the main principles behind of the proposed community platform is the use of an entity-centric abstraction to provide an uniform representation of objects, in both the real and the virtual world, that are relevant to the platform.

Entities and Entity Types

node

An entity *En* , is defined by its metadata as: *En =<id, type, Attr, Rel, S>*
Where:

- *id* : is a unique identifier (e.g., an URI).
- *type* : is the type of entity, that is, the category to which it belongs to (e.g., the entity John is of type Person).
- *Attr* : is a set of attributes composed of pairs *attr = <attrname attrvalue>* describing the properties (e.g. John? date of birth is 02/01/88) of that particular entity.
- *Rel* : is a set of relational attributes composed of pairs *rel = <relname, relvalue>* describing the entity's relations (e.g., John is friendOf Paul) with other entities.
- *S* : is a set of services that can be leveraged on that specific entity; for example, a service "send email" can be enabled on the Person entity.

The entity types encoded on the *type* property are used to define the basic attributes and services a particular type of entity will have. For example, the paper entity type defines that all instances of an entity representing a paper will have the 'abstract' attribute and the 'author' relation. Furthermore, as shown in Fig. 43 , all the entity types can be arranged into an entity type lattice, which allows the inheritance and extension of the metadata and services from the parent types (at top) to the children types (at the bottom).

As an example, consider the entity type 'Paper' (shown in Fig. 43); this type would inherit metadata and services from the types Entity, Mind Product and Document.

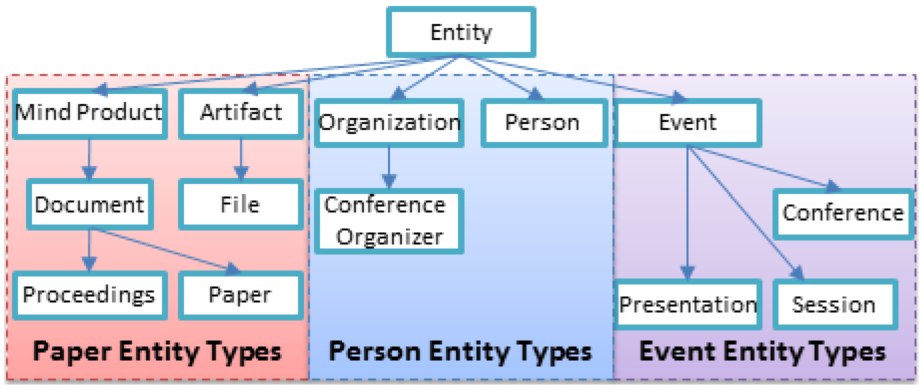


Figure 43: Entity type lattice used in the platform.

Finally, as emphasized in Fig. 43 , there are three main groupings of entity types defined on the platform. These will be discussed in more detail in the next subsection.

Papers, Persons and Events

The current version of the community platform recognizes the importance of conferences as an important axis for the scientific production and discussion between peers. This focus has helped to identify three specific entity types which are key in the current approach: Persons, the main subjects who create and discuss science; Events, how persons meet to discuss; and Papers, the artifacts created by persons for the event to motivate or as a result of discussions.

Using the entity type lattice from Fig. 43 as a guide, this subsection will offer a brief look at the three main groups of entity types, their structure and how they enable different features.

Papers

These entity types are used to represent the actual artifacts that emerge from the scientific production. Dublin Core [16] was used as the main inspiration for this specification.

The current data structures support three levels of specialization for these types of entities:

- *Mind product* : refers to any piece of intellectual work created by the human mind. Mind product is a fairly general entity type and as such has attributes like 'author' that apply to a wide range of artifacts.
- *Documents* : documents extends mind products to better represent artifacts created to transfer information or support claims. Examples of entities covered by this entity type include email, presentations and even videos.
- *Papers* : the paper entity type extends document to capture the information more specific to scientific papers (e.g. citations, keywords, publisher).

Tbl. 2 contains some examples of attributes for the entity types related to Papers.

Table 2: Attributes from the Mind Product, Document and Paper entity types.

Attribute	Entity	Description
Author	Mind Product	The set of entities that participated in the creation of the artifact (e.g. a person, a software company).
Representation	Mind Product	The set of entities of type File that contain the concrete representation (instance) of this mind product (e.g., a particular pdf file representing a document or a video clip for a given presentation).
Title	Document	A short text describing the document's subject (e.g., Applied Mathematics, A Midsummer Night's Dream).
Editor	Document	A set of the persons or organizations that edited (i.e. introduced changes or aggregated) this document.
Reference	Document	A related (but external to the document) resource that is referenced, cited, or otherwise pointed to by the document.
Version	Document	The set of documents that were versioned (i.e. are newer) from this document instance.
Abstract	Paper	A text that is related to the main topics/concepts of the paper.
Keywords	Paper	A text containing a set of words that are related to the main topics of the paper.
DBLP identifier	Paper	Digital Bibliography and Library Project
Citation	Paper	A set of documents that are cited on the paper.
Publisher	Paper	Person or Organization responsible for making the resource available.
Date of publication	Paper	Date of formal issuing by a publisher.

Persons

Persons entities represent the researchers (e.g., creators, editors, commenters, reviewers) participating in scientific activities. The inspiration of the metadata of Persons was inspired by FOAF [17] , YAGO [52] and Freebase [53] .

Tbl. 3 contains some examples of attributes for the Person entity type.

node

Table 3: Attributes from the Person entity type.

Attribute	Entity	Description
First Name	Person	The first name of a person (i.e., The name that precedes the surname).
Middle Name	Person	The middle name of a person (i.e., The name between the first name and the surname).
Last Name	Person	The last name of a person (i.e., The name used to identify the members of a family).
Title	Person	An identifying appellation signifying status or function (e.g., "Mr., Ms.").
Gender	Person	The sex of the person (male, female).
Nationality	Person	The nation of birth or naturalization.
Photo	Person	Link to a picture file that represents the person.
Email	Person	Email address used for contacting the person.
Webpage	Person	URL of the homepage or blog of the person.
Favorites	Person	Links to the preferred papers, people and events of the current person.
Organization	Person	Link to the current work/research related organization of the person.

Events

node

Events entities are used to represent organized meetings and other activities related to the scientific production of the community.

The current data structures support four levels of specialization for these types of entities:

- *Events* : refers to any general-purpose event. Normally special events like meetings, opening ceremonies and banquets are represented directly as event entities.
- *Conferences* : used to represent conferences and workshops that can, in turn, have sessions and presentations over one or more days.
- *Sessions* : represents groups of presentations, normally with a common topic.
- *Presentations* : represents each individual talk. Has links to both the person giving the talk and the corresponding paper.

Tbl. 4 contains examples of attributes for the event-related entity types.

node

Table 4: Attributes from the Event, Conference, Session and Presentation entity types.

Attribute	Entity	Description
Location	Event	Specifies where the event occurs (e.g., city, city quarter, building, set of locations, etc).
Start	Event	Start time of the event.
End	Event	End time of the event.
Organizer	Conference	Link to the person that organizes the conference.
Contact	Conference	Link to the reference person for the conference.
Moderator	Session	The person moderating or regulating the session.
Track	Session	Name or number identifying this session as one of the simultaneous sessions of the conference.
Title	Presentation	A short text describing the presentation topic.
Abstract	Presentation	A short text that is related to the main topics/concepts of the paper.
Speaker	Presentation	Link to the person/s giving the presentation.
Slides	Presentation	Link to the file of the slides used during the presentation.
Video	Presentation	Link to the video of the presentation.
Paper	Presentation	Link to the corresponding paper for the current presentation.

3. Data-Centric Services

node

Using the previously defined entity-based metadata management as a base, we define the data-centric services that operate directly on the entity metadata to offer meaningful concept-enabled functionalities.

Classification and Natural Language Processing

node

Classifications have been used for centuries to catalogue and search large sets of objects (e.g. classifying papers based on the topics they discuss). Classifications normally describe their contents by using natural language label (e.g. 'Computer Science', 'Databases').

The underlying idea of a semantic-based service is to have information encoded in a way that can be unambiguously interpreted by software agents, thus permitting them to find, share and integrate information more easily [40] . Unfortunately people normally annotate and classify documents by using ambiguous natural language, and trying to educate them to do otherwise requires them to go through a (normally considered) burdensome learning curve. A formal classification or a lightweight ontology ([41] and [42]) is, on the other hand, a classification where labels are written in a propositional concept language. As such, formal classification can be reasoned about far more easily than natural language sentences.

To obtain the best of the two worlds (i.e. the familiarity of natural language classification and the unambiguity of propositional concept language), approaches like [43] and [44] apply Natural Language Processing (NLP) to the natural language classifications to convert them into lightweight ontologies. This enables the platform to provide:

- *Document Classification* : assigning each paper to one or more categories based on their metadata. This would later be used to offer the user paper topic navigation features and also to enable better recommendation services.
- *Data Integration* : combining multiple sources of data is a non-trivial known issue ([45]) among large data and knowledge bases. This can be aided by the classification of each data source into a rooted tree and then the discovery of semantic relations that exist between these trees. This allows merging multiple data sources without introducing noise (in the form of duplicates or invalid data) which is a common issue for multiple-source digital libraries.

- *Semantic Search* : the system can find the semantic correspondence of an object or a set of objects that corresponds to a query entered by the user if the meaning associated with the object/s is more specific or equivalent to the meaning given to the query under a common sense interpretation. This is further elaborated and exemplified below.

Semantic Search

node

The semantic search functionality enhances the regular search text facilities, making possible to specifically search for any of the attributes belonging to an entity (e.g. search for the paper with the following keywords, author and/or references). Furthermore, thanks to the use a domain-specific the concept knowledge base, it allows the search results to contain concept-based matches besides the text-based ones.

For example, it allows the user to search for papers with the topic 'semantic search' and authors from 'Italian universities' and to find papers about the 'concept search' approach with authors from 'Trento university' (that is assuming that the underlying knowledge base contains all of the necessary concepts and relations to allow for such inference). A semantic search approach is used to perform the matching between these individual constraints and the entity attributes, i.e., it allows us to compute that the phrase 'concept search' has more specific meaning than phrase 'semantic search' and that Trento is a city which is located in Italy. Furthermore, faceted search is used to specify two constraints on the paper entity, namely topic: 'semantic search' and author.affiliation: 'Italian universities'.

Semantic search on individual attribute names and values is implemented by using the Concept Search approach [46] . Concept Search is an information retrieval approach which extends syntactic search with semantics in order to address the problems related to the ambiguity of natural language (e.g. the problems of polysemy and synonymy) by substituting words, when possible, with concepts. The main idea behind concept search approach is to reuse highly optimized retrieval models and data structures of syntactic search and preserve their efficiency while allowing for improved results when high-quality semantic information is provided. For instance, the semantic matching ([47] and [48]) of complex concepts, i.e. the core building block in the concept search approach, is implemented by using the inverted index technology.

Both features are specially useful for providing accurate results during exploratory search (allowing users to discover entities and knowledge that they did not know but that are interesting to them) and also for recommendation purposes (by performing a searches with parameters related to the active user).

4. User-Centric Services

node

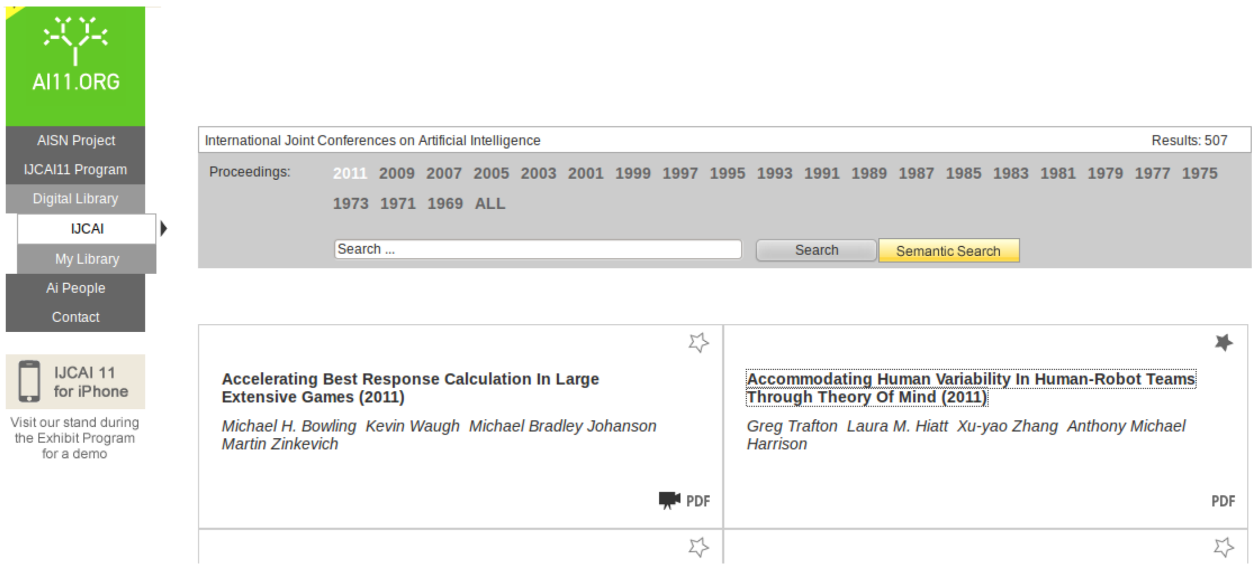
The user-centric functionalities are implemented by building highly-refined user interfaces on top of one or more of the previously defined back-end services.

Information Display and Navigation

node

The main objective of this service is use the power of the semantic services to offer a seamless and effective entity navigation interface through a web interface. For example, Fig. 44 shows a webpage (from our live demo of the platform) displaying the list of papers presented for IJCAI11.

node



International Joint Conferences on Artificial Intelligence Results: 507

Proceedings: 2011 2009 2007 2005 2003 2001 1999 1997 1995 1993 1991 1989 1987 1985 1983 1981 1979 1977 1975 1973 1971 1969 ALL

Accelerating Best Response Calculation In Large Extensive Games (2011)
Michael H. Bowling Kevin Waugh Michael Bradley Johanson Martin Zinkevich

Accommodating Human Variability In Human-Robot Teams Through Theory Of Mind (2011)
Greg Trafton Laura M. Hiatt Xu-yao Zhang Anthony Michael Harrison

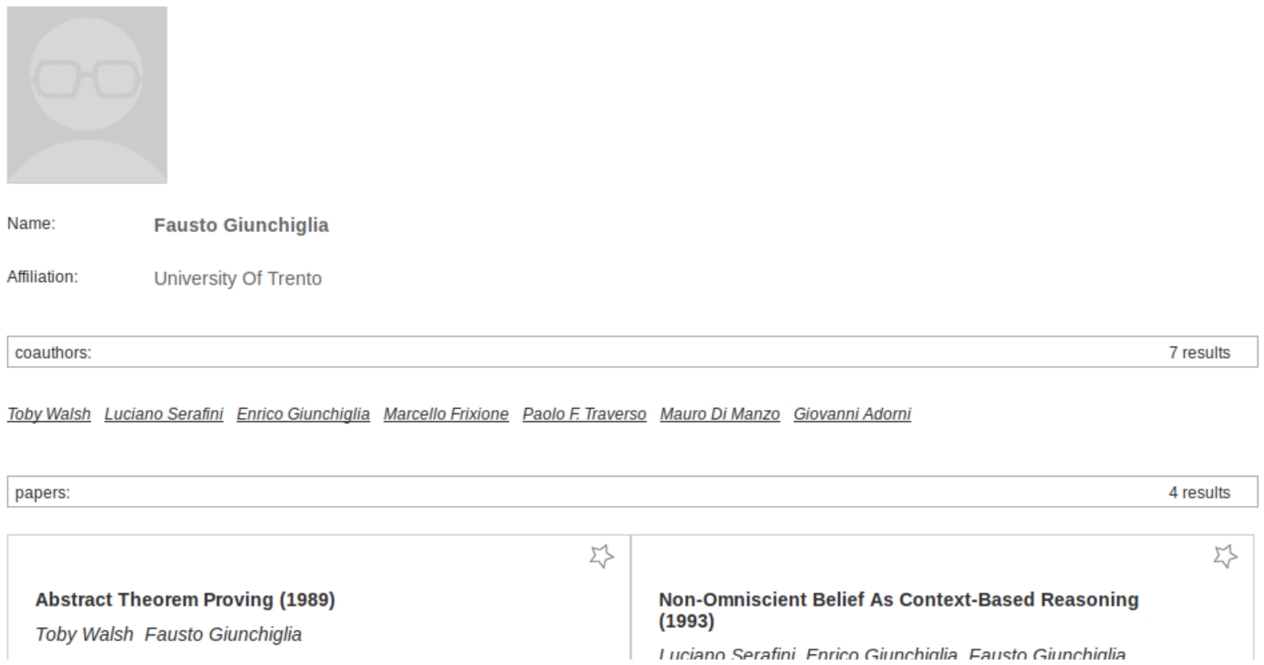
Figure 44: Paper search and browsing interface.

node

The screen showed in Fig. 44 offers several options to help the user find the information he is looking for:

- *Semantic search bar* : used to perform conventional and semantic search.
- *Entity Navigation* : the following elements in the page can be clicked:
  - *Title* : this would take the user to the paper profile page where all the metadata of the paper can be viewed and edited (provided the user has the permissions to do so).
  - *Authors* : when clicked takes the user to the Author profile page.
  - *PDF icon* : this would take the user to the actual full-text of the paper by downloading the pdf file from the IJCAI servers.
  - *Video icon* : this would take the user to a page where he can have access to a video of the that paper's presentation during IJCAI11.

Fig. 45 shows an example of a person entity profile, this displays not only the metadata related to the person (e.g. name, contact information) but also the relations that this person has with other entities in the system (e.g. co-authors, published papers). These relations are displayed as links that can be, in turn, clicked to seamlessly continue the navigation of all the entities in the platform.



Name: **Fausto Giunchiglia**

Affiliation: University Of Trento

coauthors: 7 results

Toby Walsh *Luciano Serafini* *Enrico Giunchiglia* *Marcello Frixione* *Paolo F. Traverso* *Mauro Di Manzo* *Giovanni Adorni*

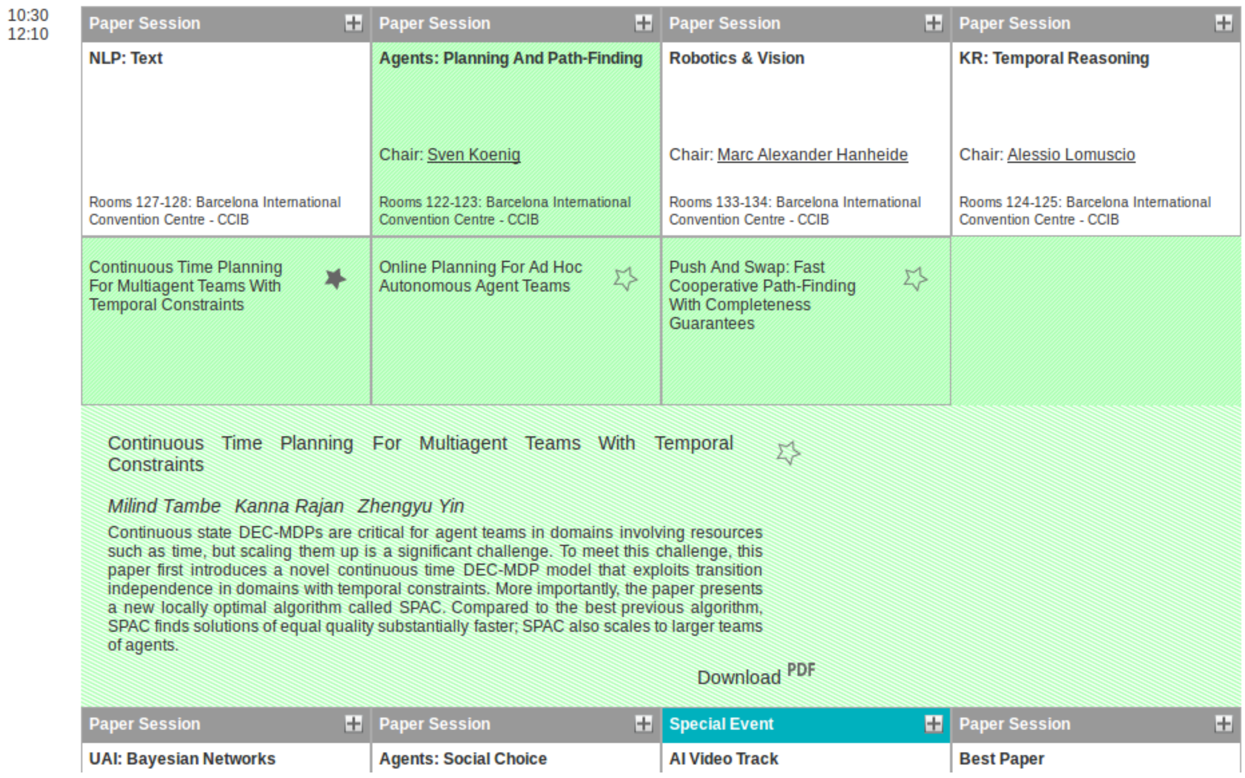
papers: 4 results

Abstract Theorem Proving (1989)
Toby Walsh Fausto Giunchiglia

Non-Omniscient Belief As Context-Based Reasoning (1993)
Luciano Serafini Enrico Giunchiglia Fausto Giunchiglia

Figure 45: Person profile example.

Fig. 46 contains an example of a screen, from the current version of the platform, displaying the events of a conference. The main interface shown in Fig. 46 is based around the idea of a expanding/contracting calendar that shows, according to the user's input, different levels of information about the conference and its sub events. Furthermore, links offering navigation to all the related entities (persons, papers, sub-events) are also made available for the user. These hundreds of interconnected entities queried and processed into rendering this webpage, are all easily findable and accessible to the user by seamless link navigation or by using the power of semantic search.



10:30 12:10	Paper Session	Paper Session	Paper Session	Paper Session
	NLP: Text	Agents: Planning And Path-Finding	Robotics & Vision	KR: Temporal Reasoning
		Chair: Sven Koenig	Chair: Marc Alexander Hanheide	Chair: Alessio Lomuscio
	Rooms 127-128: Barcelona International Convention Centre - CCIB	Rooms 122-123: Barcelona International Convention Centre - CCIB	Rooms 133-134: Barcelona International Convention Centre - CCIB	Rooms 124-125: Barcelona International Convention Centre - CCIB
	Continuous Time Planning For Multiagent Teams With Temporal Constraints	Online Planning For Ad Hoc Autonomous Agent Teams	Push And Swap: Fast Cooperative Path-Finding With Completeness Guarantees	

Continuous Time Planning For Multiagent Teams With Temporal Constraints

Milind Tambe Kanna Rajan Zhengyu Yin

Continuous state DEC-MDPs are critical for agent teams in domains involving resources such as time, but scaling them up is a significant challenge. To meet this challenge, this paper first introduces a novel continuous time DEC-MDP model that exploits transition independence in domains with temporal constraints. More importantly, the paper presents a new locally optimal algorithm called SPAC. Compared to the best previous algorithm, SPAC finds solutions of equal quality substantially faster; SPAC also scales to larger teams of agents.

Download PDF

Paper Session	Paper Session	Special Event	Paper Session
UAI: Bayesian Networks	Agents: Social Choice	AI Video Track	Best Paper

Figure 46: Current version of the platform displaying a webpage with event information.

Favourites and Personalized Entities

As a way to assist the users remembering and following their personal interests, this feature allows to define 'favourite entities' (i.e., the papers, persons and events that have a special meaning or importance for the user). As it can be noticed in Fig. 44 , Fig. 45 and Fig. 46 a small star icon appears at the top right side of each entity. Clicking on this star icon marks that entity as a favourite for the current user.

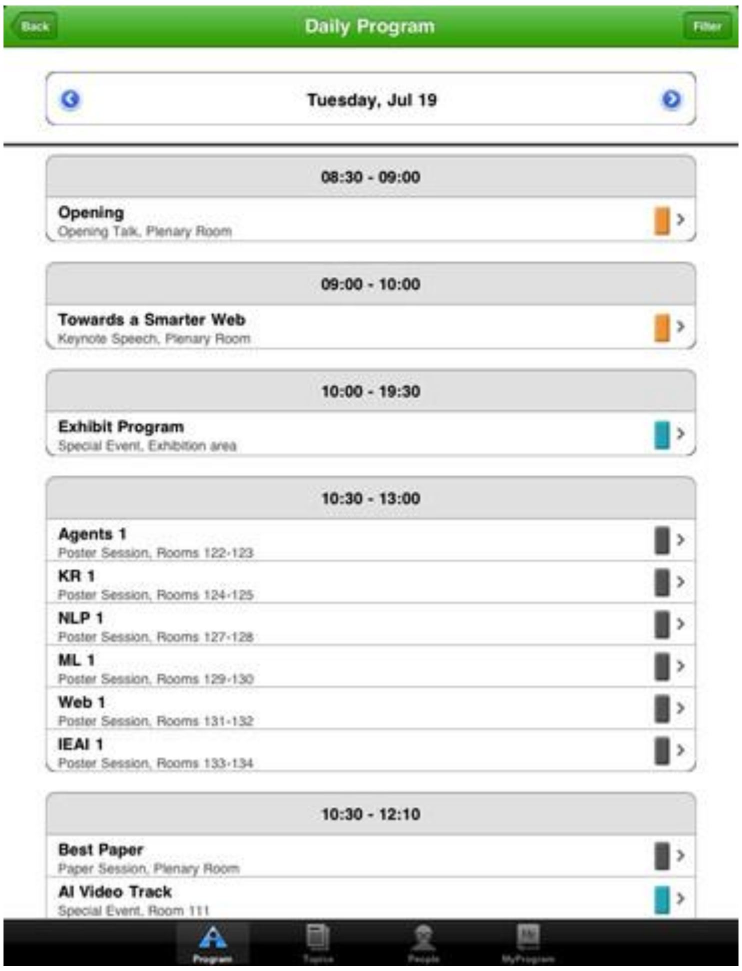
The meaning and purpose the platform gives to defining an entity as a favourite is slightly different according to the entity type:

- *My Agenda (events)* : the purpose of this page is to show a calendar with the day and time of all the favourite events of the current user.
- *My Library (papers)* : displays all the favourite papers of user in a single list, allowing for easy reference, reading of the pdf or viewing of the associated video presentation.
- *My Contacts (persons)* : shows a list of all persons marked as favourite by the current user. Future extensions include extending the My Contacts page to include and capture the different relations that may exist between researchers (e.g., is my collaborator, co-author, advisor).

Besides defining which entities are important for the user, the system also has plans to allow the user to add their own private annotations and tags to the entities. This personalization augments the ability to define favourites by also allowing the user to capture the particular reason of interest in the target entity (e.g., A user adds the 'I need to read later' annotation to paper in its favourite list and a 'I really liked the abstract' to other).

Mobile Support

Continuing with the objective of providing assistance to the members of the community, the mobile client of aims to offer key functionalities from the platform in a ubiquitous manner. More specifically, it's objective is to provide support to manage the almost chaotic event of attending to a big scientific conference like IJCAI. The mobile client is mainly focusing on managing events and helping the user build his own personalized program of the conference. Fig. 47 shows an example of how the different sub events of the conference are displayed in the mobile client.



Daily Program

Tuesday, Jul 19

08:30 - 09:00

Opening
Opening Talk, Plenary Room

09:00 - 10:00

Towards a Smarter Web
Keynote Speech, Plenary Room

10:00 - 19:30

Exhibit Program
Special Event, Exhibition area

10:30 - 13:00

Agents 1
Poster Session, Rooms 122-123

KR 1
Poster Session, Rooms 124-125

NLP 1
Poster Session, Rooms 127-128

ML 1
Poster Session, Rooms 129-130

Web 1
Poster Session, Rooms 131-132

IEAI 1
Poster Session, Rooms 133-134

10:30 - 12:10

Best Paper
Paper Session, Plenary Room

AI Video Track
Special Event, Room 111

Figure 47: Mobile client displaying the program of a conference day.

In a similar fashion to the web interface, the mobile client allows to browse the different events in a conference, having access through them to the papers (which can be comfortably read on the tablet version of the mobile client) and their authors. Thanks to these features the mobile client acts almost as a 'digital brochure' of the conference, assisting the users to make the best of their limited conference time by allowing to quickly find the talks and persons they are interested in.

5. Platform

This section will present the general architecture and some details of the most important subsystems of the proposed community platform.

The diagram at Fig. 48 shows the general architecture of the scientific community platform.

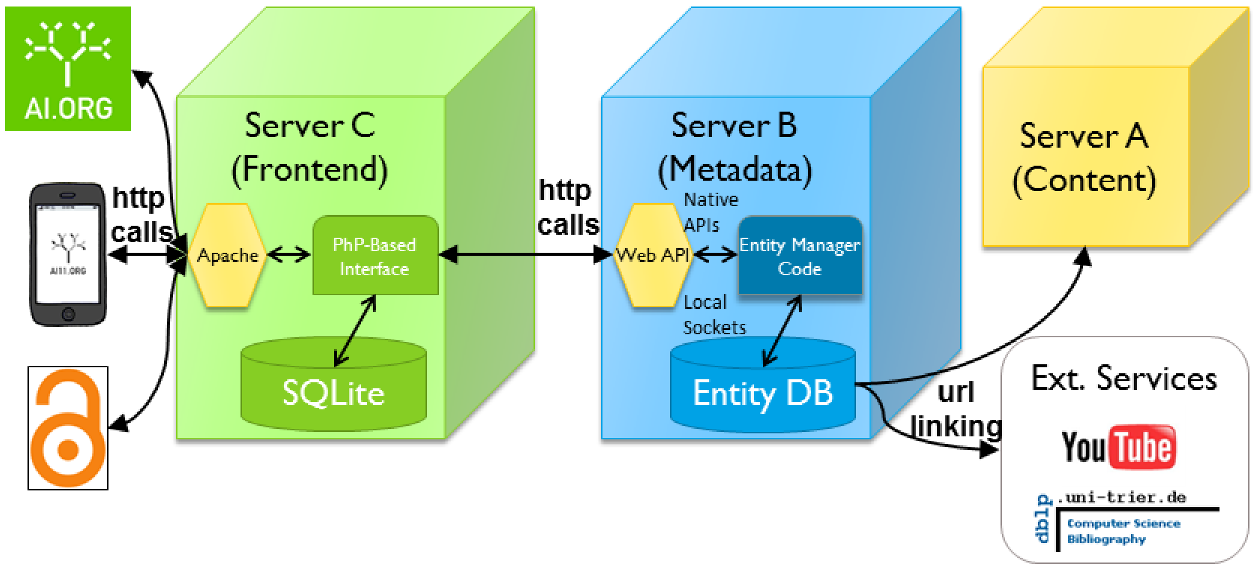


Figure 48: Community Platform General Architecture.

More specifically, in Fig. 48 , the following macro-elements may be identified:

- *Server A - Content* : the actual content (papers, presentations, etc.) are left in their original content servers and linked through URLs. In the current version Server A is managed by the IJCAI itself, it is completely independent and external to the rest of the platform. This server is also being used at the same time to serve the current IJCAI website [54] .
- *Server B - Metadata* : metadata is stored in the back-end metadata server. The server B is considered the backend of the platform as it offers both conventional and semantic-based structures management and services for the metadata. This allows the platform to avoid any copyright or licensing complications that may arise when handling and duplicating the actual content to be discussed, while still being able to refer to the actual content by the use of URLs.
- *External Services* : using the same URL linking, materials from all over the Internet (e.g. videos, profiles on other sites) can be referenced from within the platform. This external content complements the core content from Server A and their links are added to the metadata in Server B.
- *Server C* : this component mainly in charge of taking the information from the metadata server and generating the web pages/data structures that will ultimately be displayed in the clients.
- *Clients* : these are in charge of rendering forms for users and processing their inputs. Thanks to the underlying architecture, clients are allowed to be machines with a modest processing

power (e.g. mobile devices) as the more hardware-intensive processes are carried out in the servers.

There are currently three clients in development:

- *Web Client* : the browser receives enriched html pages to display and sends back the user input.
- *Mobile Client* : the mobile receives specifically formatted data structures from Server C, that the native application in the mobile device uses to build and render the pages shown to the user.
- *Open Access Client* : to comply with the Open Access Metadata Harvesting [18] Protocol, we have created a separate Open Access server that contains the subset of all the available metadata necessary to comply with this standard. By complying with this standard all of the papers managed in our platform would be indexed by the major scientific search engines.

The current architecture and the underlying specification make that the resulting platform to be easy to extend and manage both in its data-based services (e.g., extension of entities, implementation of new services) and in user-centric services (e.g., new clients like Android mobile devices easily added).

6. Conclusions

node

Our proposal integrates the information services related to Digital Libraries (papers), Social Networks (persons) and Conference-aid systems (events) into a novel platform with the objective of bridging the distance between the conventional and the web-based scientific discourse. The advantages of said coherently integrated platform include:

1. Offer improved ways for helping the users find conventional content, persons or events they are looking for (through semantic search and navigation) along with the context and relations that these may have. The specific improvements will be based on the ability to search for title, authors, keywords and even key concepts related to what the user wants to find and to aid the search based on the navigation of the "related items". Search, access to information and interaction with results, will all be integrated in the same platform and aided by it.
2. Offer a way of introducing new web-based interactions like commenting, tagging and creation relations for all the content in the site in a certified (i.e. approved and validated by the management of the site) way.
3. Provide ubiquitous/live services during conferences that, through the use of portable devices, would help attendants to find, keep track and take notes about the events happening. All this integrated into the main platform, which would both keep track of this for the user and for computing meaningful statistics about the conference. While similar services to these already exist, they are fairly localized endeavours proving a sort of "your guide during the conference" services. By integrating with the social network we want to also achieve "your preparation to the conference" and "this conference continues online" services that have yet to be coherently presented.
4. Reformat the vast volumes of legacy information into a reusable and easily citable format. This would allow the users to browse information from conference that happened over 30 years ago and also have access to information and services similar to the recent conferences (that were specifically tailored for the platform).

After a successful preview of a prototype during the IJCAI11 conference, through interviews and surveys, we got important feedback and suggestions that will be discussed in the results and conclusions part of the thesis.

2. A Layered Artifact Creation Environment

node

1. Introduction

node

Several studies state the importance of breaking down the current elements of dissemination (e.g., papers, web posts) into their data and discourse components to facilitate absorption and reuse of the actual knowledge conveyed in them, while also keeping track of the provenance information for credit attribution. For example, Nanopublications [51] uses an ontology to provide context to the core statements of a paper thus making it easier to find, curate and relate these to those of other papers. As a complement to the regular papers, another approach found in [52] proposes 'data papers' that are basically set of links pointing to the actual scientific output data of a research.

Furthermore, initiatives like Springer's The Executable Paper Grand Challenge [59] ask how can we improve the current scientific paper to enhance:

- *Executability* : making the content (graphs, table, equations) interactive to allow readers to check and experiment with the results.
- *Short and long-term compatibility* : offering all these features while remaining compatible with most of the present and future systems used for having access to it.
- *Copyright/licensing* : with no change of the current models for protecting the intellectual property of the paper (as a whole), offer different sharing and licensing options for the data and the methods used by the researcher.
- *Size and processing power* : be able to display (even in a limited matter) large files and convey procedures from large-scale computers.
- *Provenance* : support for registering and tracking of all actions performed on the paper or its components.

This shows that even publishers, often thought to be the more conservative side of the scientific process, have now begun questioning the possible evolutions to the current paper and publication models. The following are some approaches of the answers proposed for this problem:

- *Utopia Documents* [60] : utopia presents enhances pdf papers by using a curated entity set. The improvements include links to these entities and the embedding of web content for providing an interactive manipulation of data and graphics.
- *Collage* [61] : uses web technologies to create papers out of executable 'snippets' of code. These snippets can be used to encode datasets and experiments that can later be published on their own to allow their embedding on other papers.
- *SHARE* [62] : actually provides a full programming environment that is configured to perform calculations/simulation of processes and output the results in LateX. This allows two levels of sharing: the regular result-based one, but also a the sharing of data and procedures involved in the paper.

Based on these approaches and as an upcoming answer, this chapter discusses the implementation of the Layered Artifact Creation Environment (LACE) implementing the main concepts of the

Scientific Knowledge Object specification. As other layers of the SKO specification have already been validated in other applications/exercises, this prototype is particularly centered on exploring the features and services offered by the SKO's serialization layer.

2. Creation and Management Services

node

The main purpose of LACE is to greatly facilitate the creation, discussion and sharing of scientific artifacts and knowledge. To do so the current version of the prototype aims to apply the SKO specification into the following list of basic artifact-management services:

- Importing of currently existing content: input formats that can be automatically converted to the wiki format used in the platform currently include LaTeX and MS Word. This allows the researchers to quickly bootstrap the platform with all their previous produced scientific knowledge and artifacts.
- Easy-to-use and effective user interface: a simple to use and understand interface is also key for testing the prototype and also for eventually having the possibility of expanding the prototype into a commercial-grade application.
- Aggregation and reuse of components: the serialization layer management services included in the prototype platform allow for easy reuse of and inclusion of written material. This makes possible the definition of 'library of artifacts' (in direct comparison to the software libraries that are common in the software development) that allow the user to include and reuse the knowledge into several artifacts (and fully complementing the serialization layer structures).
- Internal and external references: a robust referencing system that allows to reference to sources in several ways. The current platform supports footnotes, citations, external and internal links and numbering of elements (e.g., like tables, figures, sections).
- Enrichment of artifact with navigation links: the augmenting of the final documents with live web links that can be clicked to find the different versions of component entities and also detailed metadata about them.
- Search and Navigation: display of all the available content in the platform in interactive Mindmap-like [63] graphical representations, while also providing search powered by both content and metacontent.
- Version control: basic support for creation of different versions of the same artifact and additionally support for forking/branching.
- Review-support features: basic features like localized annotations that can be used to capture review corrections, and comment threads where more elaborate discussions can take place.
- Exporting of Content: output to pdf, the current widely supported format for scientific publication, is the main supported output. Additionally output formats like LaTeX and MS Word is also partially supported for added interoperability.

3. Implementing Scientific Knowledge Objects

node

With the overall motivation and basic services to offer clear, this discusses the details of the platform used to implement the SKO specification and how each of the structures defined in it was represented and supported by this platform.

Platform Choice and Development

node

To delve right into the target set of issues of this study, as opposed to developing everything from zero, it was chosen to take an already established document creation environment and extending it to support the SKO specification. The options considered were LaTeX, Microsoft Word, Open Office and wiki-based platforms.

After considering several factors the free software platform Xwiki [64] was chosen as the base for the prototype, mainly for the following reasons:

- Open platform with source code available
- Easy extensible by using Velocity [65] and Groovy [66] that allows for Java code to be embedded directly into the content.
- Is able to export the wiki pages directly into several formats
- It allows for markup to be introduced in the text but the syntax for doing so is more minimal that the one used by LaTeX
- It can seamlessly alternate from a comfortable What you See Is What You Get (WSIWYG) visual editing environment and a raw markup code editing environment. Furthermore, direct integration with MS Word is also available which allows to edit wiki pages from the MS Word application (as shown in Fig. 49).
- It was straight-forward to code an extension for displaying the Semantic Layer Nodes as a sidebar in the creating environment
- Web display and integrating version control makes for easy collaboration and review
- Arbitrary attachments can be attached and referenced
- Links can be added to the final output of the creation environment (even on static artifacts like pdf) that when clicked would point to the different components of the SKO being managed.
- Executable applications/plug-ins/widgets can be easily attached

node

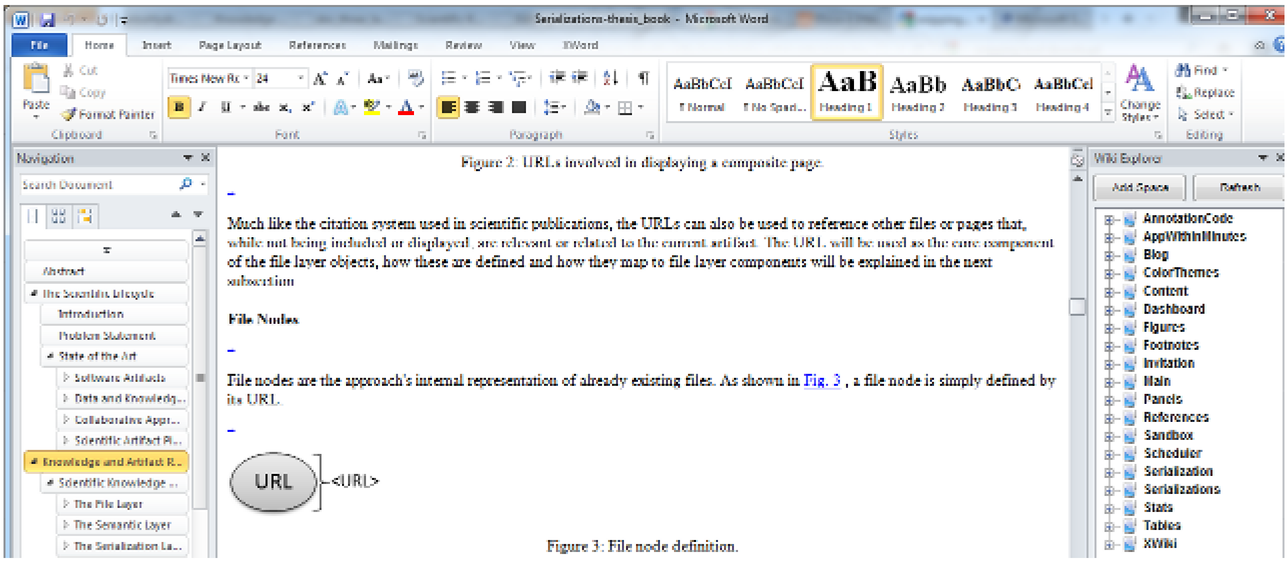


Figure 2: URLs involved in displaying a composite page.

Much like the citation system used in scientific publications, the URLs can also be used to reference other files or pages that, while not being included or displayed, are relevant or related to the current artifact. The URL will be used as the core component of the file layer objects, how these are defined and how they map to file layer components will be explained in the next subsection

File Nodes

File nodes are the approach's internal representation of already existing files. As shown in Fig. 3 , a file node is simply defined by its URL.

Figure 3: File node definition.

Figure 49: A screenshot of the MS Word client for the LACE platform

The developed platform prototype can be found and accessed through http://uk.disi.unitn.it:8100/xwiki/bin/view/Serializations/thesis_book [67]
The specific details on how the SKO specification was implemented by using Xwiki as a base will be discussed in the following sections.

File Layer Content

As defined in the SKO specification, file nodes are basically the assignment of a URL to a discrete amount of data. The wiki page-based environment, in which is each component has its own url address, made really easy to perform a URL assignment to each of the file nodes. This URL could be later be used separately to have access to the resource directly (without going through the wiki interface), as shown in Fig. 50 .

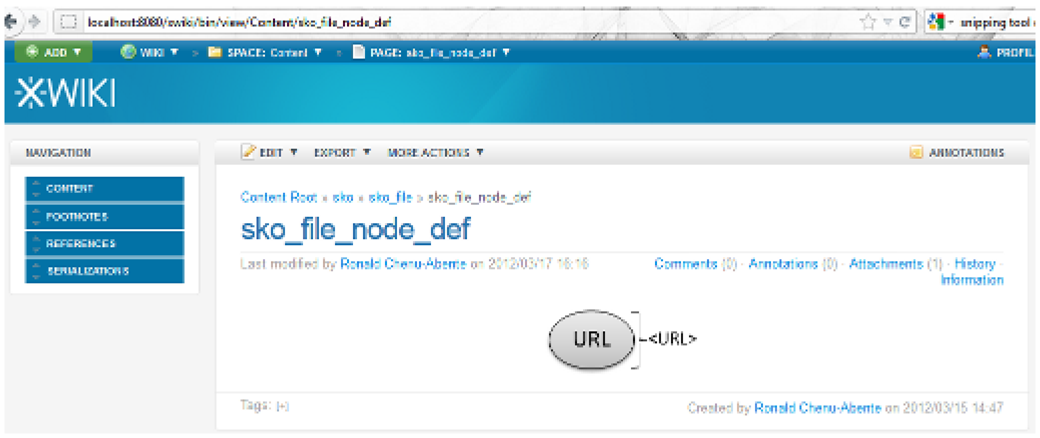

localhost:8080/xwiki/bin/view/Content/sko_file_node_def
ADD
WIKI
SPACE: Content
PAGE: sko_file_node_def
PROFIL
XWIKI
NAVIGATION
CONTENT
FOOTNOTES
REFERENCES
SERIALIZATIONS
EDIT
EXPORT
MORE ACTIONS
ANNOTATIONS
Content Root » sko » sko_file » sko_file_node_def
sko_file_node_def
Last modified by Ronald Chenu-Abente on 2012/03/17 18:16
Comments (0) · Annotations (0) · Attachments (1) · History · Information
URL
<URL>
Tags: [+]
Created by Ronald Chenu-Abente on 2012/03/15 14:47

Figure 50: The individual page of an image file node can be accessed directly by its URL

The types of files supported in the prototype version of the platform are:

- *Text* : text is stored directly as wiki pages. The Xwiki syntax allows writing complex formulas by using special symbols and macros.
- *Tables* : tables have a dedicated syntax en Xwiki that can be used to write them directly as text.
- *Images* : images are uploaded to Xwiki as attachments, these can later be included directly in the text by using a reference macro.
- *Arbitrary files* : similarly to images, arbitrary files of any format can also be uploaded as attachment and referenced from the text. However, these files do not have a visual representation like the images so they will appear as an hyperlink that can be clicked to download.

File fragments are currently only supported for the use of semantic annotations (see next section). A more system-wide implementation of file fragments would, however, be straightforward by extending an inclusion macro from the Xwiki platform to take numerical values to define fragment's boundaries.

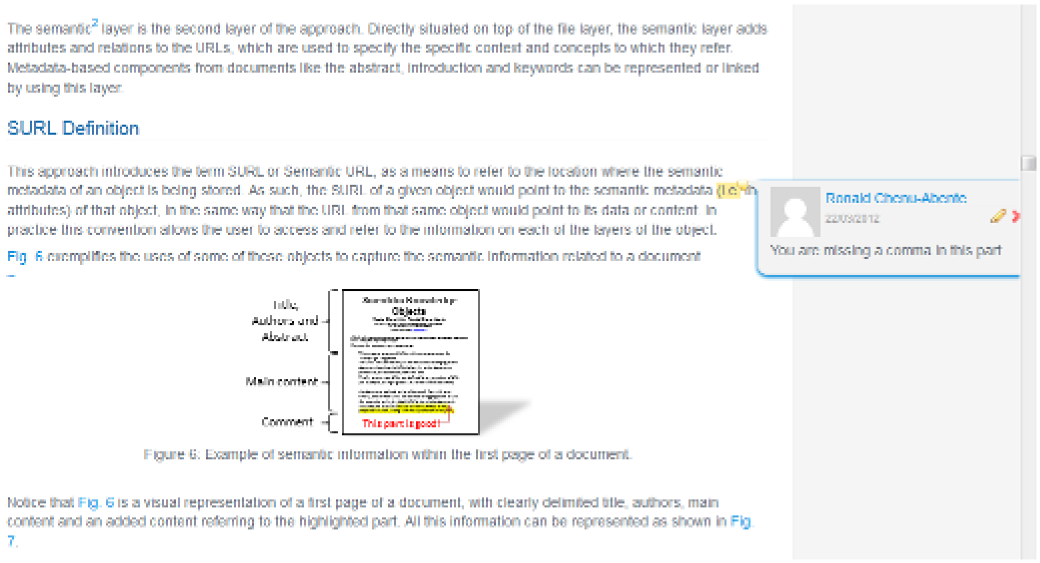
Note that up to this level the Xwiki platform is simply working as a sort of file system, storing all the file nodes and assigning them a URL that can later be used to reference them.

Semantic Layer Metadata

While the purpose of LACE prototype is not directly centered around testing the semantic layer structure (as the ijcai community application form the previous chapter was), all of the defined structures of the semantic layer are represented and partially operational.

Semantic annotations are the core structure implemented in the prototype, allowing the creation of semantic information in the form of:

- *Annotations* : Xwiki annotation functionality is accessed by clicking the annotation button at the top right of the screen. This will allow the user to select a particular part of the text and to enter text to annotate that fragment. An example of annotations and other semantic layer features is shown in Fig. 51 .
- *Version notes* : at the creation of each version, the platform allows entering text to annotate that version. This is normally used to details changes and updated with respect to the previous version.



The semantic[2] layer is the second layer of the approach. Directly situated on top of the file layer, the semantic layer adds attributes and relations to the URLs, which are used to specify the specific context and concepts to which they refer. Metadata-based components from documents like the abstract, introduction and keywords can be represented or linked by using this layer.

SURL Definition

This approach introduces the term SURL or Semantic URL, as a means to refer to the location where the semantic metadata of an object is being stored. As such, the SURL of a given object would point to the semantic metadata (i.e. the attributes) of that object, in the same way that the URL from that same object would point to its data or content. In practice this convention allows the user to access and refer to the information on each of the layers of the object.

Fig. 6 exemplifies the uses of some of these objects to capture the semantic information related to a document

Figure 6: Example of semantic information within the first page of a document.

Notice that Fig. 6 is a visual representation of a first page of a document, with clearly delimited title, authors, main content and an added content referring to the highlighted part. All this information can be represented as shown in Fig. 7.

Ronald Chenu-Abente
22/03/2012
You are missing a comma in this part

Figure 51: Example of semantic layer features implemented in the platform include annotations, links, etc.

Current implementation of the platform represents semantic nodes not as stand-alone entities (as suggested in the SKO specification), instead the semantic node is combined with its corresponding file node into a single wiki page. This partial implementation of a semantic node supports:

- *Comments* : the controls for leaving comments can be found at the end of each page. Unlike annotations that apply to a selected fragment of text, the comments apply to the whole wiki page.
- *Tags* : at the end of each wiki page, Xwiki allows the definition of natural language tags that are applied to the whole page.

- *Links* : internal (to other elements of the wiki) and external (to the Internet at large) links can be defined directly in main text by using the wiki syntax.
- *Parent attribute* : each wiki page has a parent page attribute that is used within the platform to define the parent SKO of the current semantic node.

Finally, semantic SKOs are represented as wiki pages with an empty content (i.e. text). These can, however, still have semantic annotations like comments, tags or others that when set would apply to the aggregation of all the children nodes of the SKO. Using the parent property of the pages the platform facilitates connecting the semantic SKOs and nodes to their parent SKOs, thus forming the semantic tree of the document as shown in Fig. 52

node

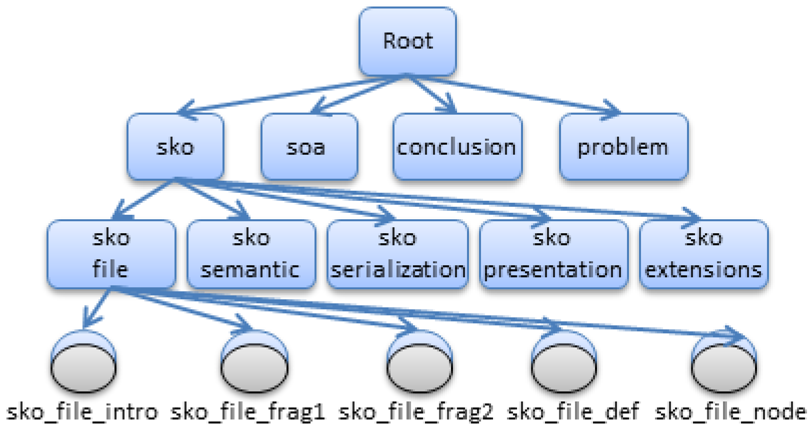


Figure 52: An excerpt of the semantic tree of this thesis. Only one branch is expanded until reaching the leaves

node

Note that in Fig. 52
the name of each semantic SKO or node is inherited from all of the parent SKOs. This is actually not required by the platform but it was chosen as a naming convention to help the users understand the structure of the semantic tree. Future versions of the platform will automatically create and display the 'parent path' so keeping a additional naming convention would not be necessary.

Since no dedicated support for bibliography is available in Xwiki, references were implemented in the current version by creating a page (i.e., a file + semantic node) containing the reference text. By using identifiers and scripts these references are introduced directly into the text where they are referred to and also at the end of the document for the actual bibliography part of the thesis. Additionally, since the conventional display of Xwiki is a continuous web-based environment and not a (more conventional for papers) paginated one, footnotes are implemented just a links to the material they refer to. As a compromise to printed versions, a bibliography-like list of footnotes is included as an appendix.

Finally, the definition of general attributes for semantic nodes and semantic SKOs is not currently implemented in the prototype but, as Xwiki support the creation of template structures that can hold attribute/value pairs, this would also be a simple extension.
Despite the limitations the current version, note that Xwiki is able to capture a wide-range of semantic information. Effectively upgrading the role of a simple file system discussed before into a semantic data management system.

Serialization Layer Metadata

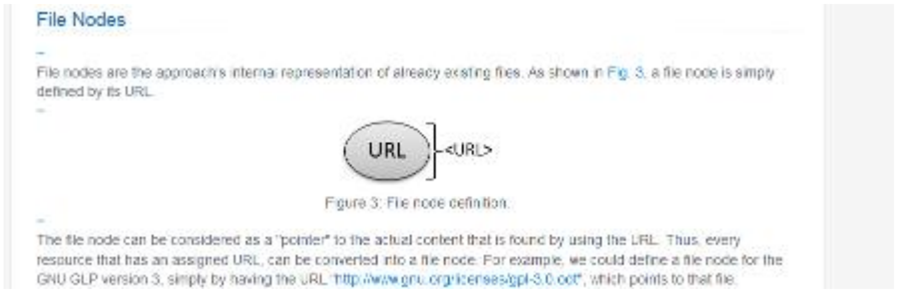
node

As previously mentioned the main purpose of the current version of the LACE prototype platform is to provide a proof-of-concept for the serialization layer of the SKO specification. To accomplish this, through scripting, the possibility of using the following macros were added to Xwiki pages:

- *include_node(URL)* : when rendering the page this macro call is replaced with the actual content of the node pointed by the argument URL. This works regardless of if what being pointed by the URL is text, tables, images or other serialization layer objects. Note that as an additional bread-crumb-like navigation, this also prints a very small link to the included object semantic node.
- *reference_node(URL)* : when rendering the page this macro is replaced by a reference number corresponding to the object pointed by the argument URL. This macro has actually 5 variants:
    - *reference_figure(URL)* : displays a link with the text "Fig X", where X is the order of appearance of the figure from the start of the document.
    - *reference_table(URL)* : displays a link with the text "Tbl X", where X is the order of appearance of the table from the start of the document.
    - *reference_footnote(URL)* : displays a superscript link with the text "X", where X is the order of appearance of the footnote from the start of the document.
    - *reference_citation(URL)* : displays a link with the text "[X]", where X is the order of appearance of the citation from the start of the document.
    - *reference_section(URL)* : displays a link with the text "Sec X", where X is the order of appearance of the section from the start of the chapter.
- include_index(URL) *: similar to include_node but instead of printing the full content of the pointed object, it prints only the titles as links.*
- include_bibliography(URL) *: calls reference_citation(URL) and then includes the full text of the citation. This is used to print th bibliography at the end of the thesis.*

Fig. 53 contains an example of the results of some of the previous macros. Note that the small links found at the top left of each content item lead to the original semantic nodes being included.

node


File Nodes

File nodes are the approach's internal representation of already existing files. As shown in Fig. 3, a file node is simply defined by its URL.

Figure 3: File node definition.

The file node can be considered as a "pointer" to the actual content that is found by using the URL. Thus, every resource that has an assigned URL, can be converted into a file node. For example, we could define a file node for the GNU GLP version 3, simply by having the URL "http://www.gnu.org/licenses/gpl-3.0.txt", which points to that file.

Figure 53: Example screen-shot showing the result of applying serialization layer structures

node

A serialization layer node is then defined as a wiki page where *only* the following elements are allowed:

1. The previously defined include and reference macros, and
   2. Section Headers

In addition, for defining a serialization layer SKO as explained in the SKO specification, the first rule is made more strict. Pages representing serialization layer SKOs are only allowed to have

1. Include macros that target only serialization layer nodes or SKOs
2. Section Header

Serialization fragments are partially implemented, a custom *include_node_section(URL, header_txt)* was also defined. The additional argument *header_txt* has to be filled with the full text of header that wants to be included. The caveat is that the granularity is limited to the header delimited fragments of a serialization (whereas this limitation does not exist on the SKO specification).

The serialization tree in Fig. 54 is an example of the serialization layer structures implemented.

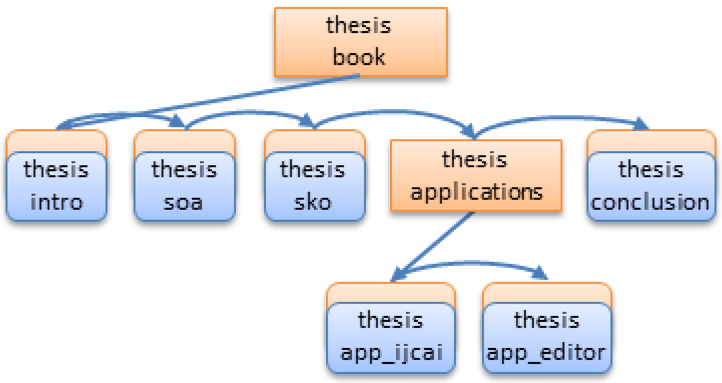


Figure 54: Full serialization tree of this thesis, expanded up to the level of serialization nodes.

As a more comprehensive example, the full script for these thesis' related serialization nodes and SKOs are included in the Appendix.

Note that with the addition of the serialization layer, the LACE prototype finally becomes a platform for creation, management and reuse of artifacts and knowledge. All the concrete services and features discussed in the previous section of this platform are enabled by this last layer.

Presentation Layer Metadata

Presentation layer objects are not currently individualized and implemented in the current version of the LACE prototype. The choice not focus on this layer is based on the fact that numerous format and style management solutions currently exist for both:

- *Local/specific presentation directives (i.e. annotations)* : this presentation layer directives are focused on a specific part of content. The Xwiki platform partially integrate these directives as part of it's syntax (e.g. italics, numbered, lists, etc.) and other artifact formats like MS Word and LaTeX have their own internal markup formats.
- *Global presentation directives (i.e. nodes and SKOs)* : this presentation layer directives define general directives that apply to the whole artifact (e.g. default font, background color). Again these are internally represented by internal markup formats but, in addition

Cascading Style Sheets [63] (CSS), is a style sheet language with great adoption that its used to specify these global look and formatting of web pages.

If the presentation is deemed important in future version of the prototype, a full support for CSS and a better individualization of the local presentation directives would be enough to cover the presentation layer of the SKO specification. The addition of this layer would enhance the platform to become a multi-format, multi-presentation creation, management and reuse environment for artifacts and knowledge.

4. Conclusions

node

The current prototype of Layered Artifact Creation Environment platform successfully implemented the all the structures and advantages from serialization layer of the SKO specification while also implementing key features from the other layers. Specially worth mentioning are:

1. Like the other approaches mentioned in the introduction, having a curated set of context relevant metadata and concept base is key to provide advanced services based on the information from the semantic layer (like classification, concept search).
2. The navigable visualization of the knowledge and artifacts in the Mindmap-like representation offered by the semantic tree and serialization tree structure offers an interesting 'concept-oriented' (i.e.,) creation approach have been found to enhance motivation motivation, understanding and recall in both the classroom and meeting room [53] .
3. The currently implemented importing procedure is limited to file layer information (e.g. text, images, tables). This means the user needs to personally annotate and reorganize his content to include information for the semantic and serialization layers. Ameliorate this problem and offer a more complete importing procedure, future versions would incorporate entity recognition ([54]) and paper pattern recognition ([55]).
4. This leads towards an overlapping area with the other previously SKO specification application. The metadata-based community platform could provide curated semantic layer information for the papers in it's entity set, while also acting as a gateway for bringing collaboration and discussion into the LACE platform.
5. With more work to include also interact-able web content into the actual artifacts, the proposed platform is suitable for responding to all of the requirements of challenges like Springer's Executable Paper.

A self-evaluation of the problems and advantages of this application will be discussed in the results and conclusions part of the thesis.

Part IV. Results and Conclusions

node

1. Analysis of Evaluation Results

node

The following contains a short description of the two valuation processes performed as part of this study and its obtained results.

1. A Metadata-Based Community Platform

node

To obtain feedback and validation for the Metadata-Based Platform we named it as the AI Social Network (AISN) project and set up a booth in the IJCAI11 congress that took place in Barcelona from July 16-22 2001. The interested persons approaching the booth were given a quick explanation of the current platform, invited to try a couple of the features in provided web and the mobile clients, and finally asked to fill a quick questionnaire. In total the information, feedback and suggestions of 71 IJCAI11 attendees willing to contribute with the project were collected.

In general, the following points where obtained as a conclusion from this exercise:

- The initial reaction to the use of the platform was largely positive. Many persons were curios and optimistic about having a system to bring AI researchers together and that brings better access to the past IJCAI papers.
- The name we chose to introduce the system (i.e. AI Social network) was initially met with distrust as the 'Social Network' name is normally associated with Facebook and thus it is associated a non-research-related applications
- People expressed their interest in the features offering presentation videos and slides, as it could aid them to update themselves on the presentations they have missed. The same interest was not expressed when offered the possibility of live streaming the presentations.
- Privacy is a very important concern for the users of these types of services. If they think that their private information is at risk they will choose to not participate.
- The participants expressed interest in having access to and navigating coauthor and citation networks.
- Finally the most successful feature was the event-based agenda that synchronized the favorites events of the user across devices.

All in all, the amount and the positivity of results obtained during this validation exercise were promising. Specially when considering that the version previewed IJCAI11 was only a beta-level prototype, that we had a small window of time to have user test the system in the busy environment of the conference and, that the main objective was getting feedback for the future direction of the platform.

2. A Layered Artifact Creation Environment

node

To validate the LACE platform the author of this thesis used it to convert a great part of his PhD's scientific production into SKOs. He then proceeded to use the prototype to write new content specifically aimed at this thesis. Finally, by using serialization layer structures he assembled this thesis book by using new and reused resources from his previous work.

The following is a self-evaluation on the LACE platform performed by the author:

- Pros of using the LACE platform
    - Content reuse. A great quantity of nodes (specially figures, tables and text for the description of the specification parts) were reused across more than 2 documents.
    - Easy correction of source and management of forks/branches. If the changed wanted to be implemented in all places that reused that content, the author would edit the original file + serialization node. Otherwise, it was also possible to add changes to the specific instance of the content, effectively forking form the original source.
    - Mind map-like content creation: the interactive semantic tree and serialization tree structures accessible in the left bar of the platform allows to quickly find the information you are looking for according to its semantics or the part of the document it belongs to.
    - Easy to use web interface that offers both a visual and code editor. Furthermore, connector extension for MS Word is also available which allows to create content for LACE directly from the MS Word interface.
    - Collaborative potential, features like simultaneous web access, annotations and comments; allow for collaboration within the creation environment (though this feature was not extensively tested as the author is the only direct contributor to this thesis).
- Cons of using the LACE platform
    - Importing only file layer information, while having to manually fill all the other layers. This process is time consuming and most likely cannot be expected of regular users.
    - Requires manual scripting and knowledge of the specification for its correct use. The ideal would be that all of these details are abstracted away by intuitive user interfaces.

Furthermore, additional results obtained from studying the data produced by the creation of this thesis suggest that:

- While introduction and conclusion content tend not to be much reused, the actual theory and state of the art parts are almost always reused (at least in a partial manner).
- Non-text content, like tables and figures are reused more frequently (these most likely contain either the description of the methods, models, examples or (in other papers) experimental data.

More detailed results and scripts produced during this process may be found in the Appendix.

2. Possible Extensions and Future Work

node

- Greatly diminish the barrier for adoption by implementing importing procedures for each of the layers of the specification:
    - Semantic Layer importing: supporting common and upcoming scientific metadata specifications (like Open Access Metadata Harvesting Protocol, micro formats, etc.) and ontologies would greatly improve metadata interoperability. Furthermore, procedures for extracting semantic layer metadata [54] directly from the file layer (pdfs) are also being implemented for legacy papers.
    - Serialization Layer importing: being able to identify and detect the discourse patterns in the papers ([55]) would greatly help towards developing an algorithm for extracting serialization layer information from the file and semantic layers.

- Improve the end-user experience of the LACE platform by offering services and interactions that facilitate the management and navigation of object in each of the layers, while offering useful services based on the available semantic and serialization information (e.g., heading-text-folding, recommendation of citations and discuss patterns based on already written text).
- Explore the licensing and credit attribution options that can be offered in the platform (as these issues are considered very important by the users)
- Expand the types of artifact that the LACE platform can create from papers to presentations, meeting notes, blog posts, emails, etc.
- A new release of the AI community platform is currently in planning stages. Some of the discussed features are: crowdsourcing features used to improve the accuracy of the information in the platform, more social interaction features, creation of categorizations of the sub fields in the AI field (to facilitate finding papers relevant to particular subjects) and more fine-grained organizations and affiliations. (i.e., supporting the differentiation between university, department, research group).
- Explore possible integrations between the LACE platform and the AI community platform:
  - Use the curated entity set from the AI community platform to support entity and metadata recognition in LACE (e.g., recognize the name of an author being written as text in LACE and add a link to the profile of that author in the community platform).
  - Allow the community that has access to AI community platform to discuss existing papers and collaborate in the creation of new ones by using LACE.
- Finish the formalization of the SKO specification into a SKO model

3. General Conclusion

node

The scientific community at large is in the middle of a transition between the original scientific processes from the 17th century and the web and semantic-enabled world. Scientific content creators, consumers and publishers drive forward this change as a means to make the scientific process more effective and more suitable to satisfy their needs and the ones of science as a whole.

The Scientific Knowledge Object specification was created as a step forward in this transition and to answer to the call for the evolution made by the actors of the scientific process. The specification recognized the importance of a clear separation between: i. the content, ii. the semantics and concepts related to it, iii. the different serial orderings that may be applied to better adjust to specific dissemination venues, and iv. finally the visual presentation styles applied to it. Nevertheless, the importance of joining and inter-operating between the different formats that exist in each of this layer was also not dismissed.

The first direct application of the SKO specification is focused on acquiring, managing and curating of content and metadata. Using this information to offer services that better capture the great amounts of information produced during almost any activity of the scientific life-cycle. An early version of this partial application already has been well-received for the useful (if still somehow basic) services that offers to its users and promises, in it's next iteration, to extend its offered services well beyond the current state of the art of similar approaches.

The second application of the SKO specification mainly focused on facilitating the complex process of creation and evolution of scientific artifacts. But, above all, demonstrating the deep impact that the separation of meaning and structure during creation process for organizing the scientific production into intuitive semantic/structural maps and for promoting the reuse and sharing of 'knowledge libraries'. These knowledge libraries contain data, methods, models, figures, tables and even texts that are prepared to be included in other artifacts (mirroring the role of code libraries from the software world).

Our objective with this line of research was aimed to truly propose a bridge, a next step in the seemingly unavoidable evolution of the scientific process as a whole from physical to digital and from static/discrete to live and constantly evolving.

4. Summary of Scientific Contribution

node

Considering on the foundations set by the collaboration work on the state of the art for scientific publishing and artifacts from [56] and the identified improvement needs ([57]) , the main contribution of this work is the proposal, ([58]) application and validation ([59] and the work on the creation environment presented in this thesis) of the SKO specification and its different possible related uses ([60] , [61] , [62]).

Part V. Appendix

node

1. LACE Thesis-Writing Details

node

node

As an example of the application of the Layered Artifact Creation Environment (LACE), this appendix includes the full detailing of the serialization and semantic structures created for this thesis.

1. Serialization Script Commands

node

The following are the script commands used in this appendix to represent the serialization script used for assembling this thesis document.

- *include(name)* : includes the full content a node or SKO from the semantic or serialization layers in the text of the document.
- *figure(name, caption)* : includes the full content of a node (normally a picture) and a centered caption after the picture. This script command is also used to keep track of figure numbers and references.
- *table(name, caption)* : similarly to the previous, this includes the content of a node (normally table) with a centered caption before it.
- *header(level, text)* : includes the header text passed to the function formatted according to the numeric level argument.
- *include_titles(name)* : includes only the title and page number of a given serialization layer object and of all the sub-objects that are included in it. Generally used to produce indexes.
- *include_references(name)* : includes only the reference text of a given serialization layer object and of all the sub-objects that are included in it. Generally used to produce bibliography sections. Note that the current version of the platform reprensents reference text as non-structured semantic nodes.
- *include_footnotes(name)* : includes only the footnote number and text of a given serialization layer object and of all the sub-objects that are included in it.
- *include_figure_titles(name)* : includes only the figure number, caption and page number of all the figures included in a given serialization layer object and its sub-objects. Generally used to produce figure indexes.
- *include_table_titles(name)* : similar to the previous but concerning tables instead of figures.

2. Serialization Nodes

node

thesis_intro

```
# header(1, Abstract)
#include(Content.intro_abstract)
# header(1, Executive Summary)
#include(Content.intro_executivesum)
```

thesis_soa

```
# header(2, Introduction)
# include(Content.soa_intro)
# header(2, Problem Statement)
# include(Content.soa_problem)
# header(2, State of the Art)
# include(Content.soa_chapterintro)
# header(3, Software Artifacts)
# include(Content.soa_sw_intro)
# header(4, Software Development)
# include(Content.soa_sw_dev)
# header(4, Version control systems)
# include(Content.soa_sw_version)
# header(4, Online Software Development Services)
# include(Content.soa_sw_onlinedev)
# header(3, Data and Knowledge Management)
# include(Content.soa_kwman_intro)
# header(4, Metadata)
# include(Content.soa_kwman_metadata)
# header(4, Hypertext)
# include(Content.soa_kwman_hypertext)
# header(4, Markup Language)
# include(Content.soa_kwman_markup)
# header(4, Document Management)
# include(Content.soa_kwman_docman)
# header(3, Collaborative Approaches)
# include(Content.soa_colab_intro)
# header(4, Online Social Networks)
# include(Content.soa_colab_sns)
# header(4, Computer Supported Cooperative Work)
# include(Content.soa_colab_cscw)
# header(3, Scientific Artifact Platforms and Semantic Web for Research)
# include(Content.soa_sciweb_intro)
# header(4, ABCDE Format)
# include(Content.soa_sciweb_abcde)
# header(4, ScholOnto)
# include(Content.soa_sciweb_scholonto)
# header(4, SWRC)
# include(Content.soa_sciweb_swrc)
# header(4, SALT)
# include(Content.soa_sciweb_salt)
```

thesis_sko

```
# header(2, Scientific Knowledge Object Specification)
# include(Content.sko_intro1)
# figure(Content.sko_four_layers, The different layers of the SKO specification.)
# include(Content.sko_intro2)
# header(3, The File Layer)
# include(Content.sko_file_intro)
# header(4, URL Definition)
# include(Content.sko_file_url1)
# figure(Content.sko_file_url_def, URLs involved in displaying a composite page.)
# include(Content.sko_file_url2)
# header(4, File Nodes)
# include(Content.sko_file_node1)
# figure(Content.sko_file_node_def, File node definition.)
# include(Content.sko_file_node2)
# header(4, File Fragment Definition)
# include(Content.sko_file_frag1)
# figure(Content.sko_file_frag_ex, Text fragment example, the dashed lines show the selected part.)
# include(Content.sko_file_frag2)
# figure(Content.sko_file_frag_def, File fragment definition.)
# include(Content.sko_file_frag3)
# header(3, The Semantic Layer)
# include(Content.sko_semantic_intro)
# header(4, SURL Definition)
# include(Content.sko_semantic_surl1)
# figure(Content.sko_semantic_surl_ex, Example of semantic information within the first page of a
document.)
# include(Content.sko_semantic_surl2)
# figure(Content.sko_semantic_surl_rep, Representation of the semantic metadata of the first page
example.)
# include(Content.sko_semantic_surl3)
# header(4, Semantic-layer SKOnodes)
# include(Content.sko_semantic_node1)
# figure(Content.sko_semantic_node_def, Semantic SKOnode definition.)
# include(Content.sko_semantic_node2)
# header(4, Semantic Annotations)
# include(Content.sko_semantic_ann1)
# figure(Content.sko_semantic_ann_def, Semantic annotation definition.)
# include(Content.sko_semantic_ann2)
# header(4, Semantic-layer SKOs)
# include(Content.sko_semantic_sko1)
# figure(Content.sko_semantic_sko_ex, Example of semantic information within the full
document.)
# include(Content.sko_semantic_sko2)
# figure(Content.sko_semantic_sko_rep, Representation of the semantic information of the full
document example.)
# include(Content.sko_semantic_sko3)
# figure(Content.sko_semantic_sko_def, Semantic SKO definition.)
# include(Content.sko_semantic_sko4)
# figure(Content.sko_semantic_sko_tree, Semantic tree example.)
# include(Content.sko_semantic_sko5)
# header(3, The Serialization Layer)
# include(Content.sko_serialization_intro)
# header(4, LURL definition)
# include(Content.sko_serialization_lurl1)
# figure(Content.sko_serialization_lurl_ex, Example of the serialization information within the first
page of a document.)
# include(Content.sko_serialization_lurl2)
# figure(Content.sko_serialization_lurl_rep, Representation of the serialization information within
the first page of a document.)
# include(Content.sko_serialization_lurl3)
# header(4, Serialization-layer SKOnodes)
# include(Content.sko_serialization_node1)
# figure(Content.sko_serialization_node_def, Serialization SKOnode definition.)
# include(Content.sko_serialization_node2)
# figure(Content.sko_serialization_node_ex, Example of different serialization SKOnodes.)
# header(4, Serialization-layer Fragment Definition)
# include(Content.sko_serialization_frag1)
# figure(Content.sko_serialization_frag_def, Serialization fragment definition.)
# include(Content.sko_serialization_frag2)
# figure(Content.sko_serialization_frag_ex, Serialization fragment example.)
# include(Content.sko_serialization_frag3)
# header(4, Serialization-layer SKOs)
# include(Content.sko_serialization_sko1)
# figure(Content.sko_serialization_sko_ex, Example of aggregation of serialization SKOnodes.)
# include(Content.sko_serialization_sko2)
# figure(Content.sko_serialization_sko_rep, Representation of the aggregation of serialization
SKOnodes.)
# include(Content.sko_serialization_sko3)
# figure(Content.sko_serialization_sko_def, Serialization-layer SKO definition.)
# include(Content.sko_serialization_sko4)
# figure(Content.sko_serialization_sko_tree, Serialization tree example.)
vinclude(Content.sko_serialization_sko5)
# header(3, The Presentation Layer)
# include(Content.sko_presentation_intro)
# header(4, PURL Definition)
# include(Content.sko_presentation_purl)
# header(4, Motivation)
# include(Content.sko_presentation_motivation1)
# figure(Content.sko_presentation_purl_ex, Example of the same document in two different styles.)
# include(Content.sko_presentation_motivation2)
# figure(Content.sko_presentation_purl_rep, Representation of the presentation information for the
'Style2' document.)
# include(Content.sko_presentation_motivation3)
# header(4, Presentation-layer SKOnodes)
# include(Content.sko_presentation_node1)
# figure(Content.sko_presentation_node_def, Presentation SKOnode definition.)
# include(Content.sko_presentation_node2)
# header(4, Presentation Annotation)
# include(Content.sko_presentation_ann1)
# figure(Content.sko_presentation_ann_def, Presentation annotation definition.)
# include(Content.sko_presentation_ann2)
# header(4, Presentation-layer SKOs)
# include(Content.sko_presentation_sko1)
# figure(Content.sko_presentation_sko_def, Presentation SKO definition.)
# include(Content.sko_presentation_sko2)
# figure(Content.sko_presentation_tree, Presentation tree example.)
# include(Content.sko_presentation_sko3)
# header(2, Features and Services Enabled by the SKO Specification)
# include(Content.sko_features_intro)
# header(3, Version Control and Branching)
# include(Content.sko_features_version_intro)
# header(4, Version Annotations)
# include(Content.sko_features_version_ann1)
# figure(Content.sko_services_version_ex, A simple version contro evolution example.)
# include(Content.sko_features_version_ann2)
# figure(Content.sko_services_version_rep, Representation of the previous version control example.)
# include(Content.sko_features_version_ann3)
# figure(Content.sko_services_version_splitmerge_ex, A simple branch/fork and merge example.)
# include(Content.sko_features_version_ann4)
# figure(Content.sko_services_version_splitmerge_rep, Representation of the previous branch/fork and merge example.)
# include(Content.sko_features_version_ann5)
# header(4, Layered Version Control)
# include(Content.sko_features_version_layers1)
# figure(Content.sko_services_version_execution_rep, Example representation using the "is execution" relation.)
# include(Content.sko_features_version_layers2)
# header(3, Search and Navigation)
# include(Content.sko_features_searchnav_intro)
# header(4, Search options)
# include(Content.sko_features_searchnav_sopts)
# header(4, Navigation options)
# include(Content.sko_features_searchnav_nopts)
# header(3, Ownership and Credit Attribution)
# include(Content.sko_features_credit)
# header(3, Licensing and Copyright)
# include(Content.sko_features_intproperty)
# header(2, Possible Extensions of the SKO Specification)
# include(Content.sko_extensions_intro)
# header(3, Maturity Measure (States))
# include(Content.sko_extensions_states_intro1)
# figure(Content.sko_extensions_states_diagram, SKO states diagram.)
# include(Content.sko_extensions_states_intro2)
# table(Content.sko_extensions_states_table, Main properties of the three SKO states.)
# include(Content.sko_extensions_states_intro3)
# header(4, The Gas State)
# include(Content.sko_extensions_states_gas)
# header(4, The Liquid State)
# include(Content.sko_extensions_states_liquid1)
# figure(Content.sko_extensions_states_exdiag1, A liquid version is released to obtain feedback from friends.)
# include(Content.sko_extensions_states_liquid2)
# figure(Content.sko_extensions_states_exdiag2, The feedback is used to evolve the original document in progress document.)
# include(Content.sko_extensions_states_liquid3)
# figure(Content.sko_extensions_states_exdiag3, A liquid version is sent to the chair of a conference and gets reviewed.)
# include(Content.sko_extensions_states_liquid4)
# figure(Content.sko_extensions_states_exdiag4, The review feedback is used to make correction and evolve the original document.)
# include(Content.sko_extensions_states_liquid5)
# header(4, The Solid State)
```

# include(Content.sko_extensions_states_solid1)
# figure(Content.sko_extensions_states_exdiag5, A new liquid version is sent to the conference chair. The chair approves and releases the certified solid version.)
# include(Content.sko_extensions_states_solid2)
# header(3, Version Control for Aggregation)
# include(Content.sko_extensions_versionagg_intro)
# header(4, Changes on aggregated objects)
# include(Content.sko_extensions_versionagg_changes1)
# figure(Content.sko_extensions_aggvc_autofork_fig, Auto-fork example: as soon as a modification is introduced for the pointed component, a fork is done by the system to preserve the information being aggregated.)
# include(Content.sko_extensions_versionagg_changes2)
# figure(Content.sko_extensions_aggvc_autoupdate_fig, Auto-update example: as soon as a new version of the component appears, the system changes the "part of" annotation to point to it.)
# include(Content.sko_extensions_versionagg_changes3)
# header(4, Changes on aggregating objects)
# include(Content.sko_extensions_versionagg_changes4)

thesis_app_editor

# header(3, Introduction)
# include(Content.apps_editor_intro)
# header(3, Creation and Management Services)
# include(Content.apps_editor_services)
# header(3, Implementing Scientific Knowledge Objects)
# include(Content.apps_editor_implementation_intro)
# header(4, Platform Choice and Development)
# include(Content.apps_editor_implementation_plat1)
# figure (Content.apps_editor_xword_ss, A screenshot of the MS Word client for the LACE platform)
# include(Content.apps_editor_implementation_plat2)
# header(4, File Layer Content)
# include(Content.apps_editor_implementation_file1)
# figure (Content.apps_editor_figure_alone_ss, The individual page of an image file node can be accessed directly by its URL)
# include(Content.apps_editor_implementation_file2)
# header(4, Semantic Layer Metadata)
# include(Content.apps_editor_implementation_semantic1)
# figure (Content.apps_editor_sfeatures_ss, Example of semantic layer features implemented in the platform include annotations, links, etc.)
# include(Content.apps_editor_implementation_semantic2)
# figure (Content.apps_editor_semtree_ss, An excerpt of the semantic tree of this thesis. Only one branch is expanded until reaching the leaves)
# include(Content.apps_editor_implementation_semantic3)
# header(4, Serialization Layer Metadata)
# include(Content.apps_editor_implementation_serialization1)
# figure (Content.apps_editor_figure_content_ss, Example screen-shot showing the result of applying serialization layer structures)
# include(Content.apps_editor_implementation_serialization2)
# figure (Content.apps_editor_sertree_ss, Full serialization tree of this thesis, expanded up to the level of serialization nodes.)

# include(Content.apps_editor_implementation_serialization3)
# header(4, Presentation Layer Metadata)
# include(Content.apps_editor_implementation_presentation)
# header(3, Conclusions)
# include(Content.apps_editor_conclusions)

thesis_app_ijcai

# include(Content.apps_ijcai_abstract)
# header(3, Introduction)
# include(Content.apps_ijcai_intro)
# header(3, Entities)
# include(Content.apps_ijcai_entities_intro)
# header(4, Entities and Entity Types)
# include(Content.apps_ijcai_entities_types1)
# figure (Content.apps_ijcai_entities_etypes_lattice, Entity type lattice used in the platform.)
# include(Content.apps_ijcai_entities_types2)
# header(4, Papers, Persons and Events)
# include(Content.apps_ijcai_entities_papers_intro)
# header(5, Papers)
# include(Content.apps_ijcai_entities_papers1)
# table (Content.apps_ijcai_entities_papers_typetable, Attributes from the Mind Product, Document and Paper entity types.)
# header(5, Persons)
# include(Content.apps_ijcai_entities_persons1)
# table (Content.apps_ijcai_entities_persons_typetable, Attributes from the Person entity type.)
# header(5, Events)
# include(Content.apps_ijcai_entities_events1)
# table (Content.apps_ijcai_entities_events_typetable, Attributes from the Event, Conference, Session and Presentation entity types.)
# header(3, Data-Centric Services)
# include(Content.apps_ijcai_services_data_intro)
# header(4, Classification and Natural Language Processing)
# include(Content.apps_ijcai_services_data_classif)
# header(4, Semantic Search)
# include(Content.apps_ijcai_services_data_ssearch)
# header(3, User-Centric Services)
# include(Content.apps_ijcai_services_user_intro)
# header(4, Information Display and Navigation)
# include(Content.apps_ijcai_services_user_nav1)
# figure (Content.apps_ijcai_services_data_snav_paper_ss, Paper search and browsing interface.)
# include(Content.apps_ijcai_services_user_nav2)
# figure (Content.apps_ijcai_services_data_snav_person_ss, Person profile example.)
# include(Content.apps_ijcai_services_user_nav3)
# figure (Content.apps_ijcai_services_data_snav_event_ss, Current version of the platform displaying a webpage with event information.)
# header(4, Favourites and Personalized Entities)
# include(Content.apps_ijcai_services_user_favs)
# header(4, Mobile Support)
# include(Content.apps_ijcai_services_user_mob1)
# figure (Content.apps_ijcai_services_user_mobile_ss, Mobile client displaying the program of a

conference day.)
# include(Content.apps_ijcai_services_user_mob2)
# header(3, Platform)
# include(Content.apps_ijcai_platform1)
# figure (Content.apps_ijcai_platform_arch, Community Platform General Architecture.)
# include(Content.apps_ijcai_platform2)
# header(3, Conclusions)
# include(Content.apps_ijcai_conclusions)

thesis_conclusion

# header(2, Analysis of Evaluation Results)
# include(Content.conclusions_results_intro)
# header(3, A Metadata-Based Community Platform)
# include(Content.conclusions_results_mbcp)
# header(3, A Layered Artifact Creation Environment)
# include(Content.conclusions_results_lace)
# header(2, Possible Extensions and Future Work)
# include(Content.conclusions_future)
# header(2, General Conclusion)
# include(Content.conclusions_general)
# header(2, Summary of Scientific Contribution)
# include(Content.conclusions_contribution)

appendix_classroom

# include(Content.classroom_intro1)
# figure(Content.apps_lb_arch, Different elements of the LiquidBook exercise.)
# include(Content.classroom_intro2)
# header(3, Execution of the exercise)
# include(Content.classroom_exercise)
# header(3, Preliminary Results)
# include(Content.classroom_premresults)
# header(3, Conclusions)
# include(Content.classroom_conclusions)

appendix_events

# include(Content.events_intro)
# header(3, Entities)
# include(Content.events_entities)
# header(3, Events)
# include(Content.events_definition)
# header(3, Events Compositionality)
# include(Content.events_composing_intro)
# header(4, Restrictions on Time t)
# include(Content.events_composing_time1)
# figure(Content.sko_events_timeline, An example event timeline separated into tree-like sub events.)
# include(Content.events_composing_time2)
# header(4, Restriction on Context)

# include(Content.events_composing_context)
# header(4, Restriction on Linked Events)
# include(Content.events_composing_linked)
# header(3, Conclusions)
# include(Content.events_conclusions)

appendix_inclusion

# include(Content.inclusion_intro)
# header(3, Problematic)
# include(Content.inclusion_problem)
# header(3, Contributions of a SKO-based Platform)
# include(Content.inclusion_contribution)

appendix_lace_thesis

# include(Content.apps_editor_fullserial_intro)
# header(3, Serialization Script Commands)
# include(Content.apps_editor_fullserial_commands)
# header(3, Serialization Nodes)
# include(Content.apps_editor_fullserial_nodes)
# header(3, Serialization SKOs)
# include(Content.apps_editor_fullserial_skos)
# header(3, Full Semantic and Serialization Trees)
# include(Content.apps_editor_fullserial_tree)
# include(Content.apps_editor_fullserial_semtree)

appendix_misc

# header(3, Metadata vs Semantic)
# include(Content.sko_semantic_vsmetadata1)
# figure(Content.sko_semantic_vsmetadata_pic1, The second to fourth layers from the original SKO layer separation are all metadata.)
# include(Content.sko_semantic_vsmetadata2)
# figure(Content.sko_semantic_vsmetadata_pic2, The SKO specification separates the metadata into three semi-independent sub-layers.)
# include(Content.sko_semantic_vsmetadata3)

3. Serialization SKOs

node

thesis_book

# include_titles(Serializations.thesis_core)
# include_figure_titles(Serializations.thesis_core)
# include_table_titles(Serializations.thesis_core)
# include(Serializations.thesis_core)
# include_references(Serializations.thesis_core)

thesis_core

```
# header(1, Abstract)
# include(Serializations.thesis_intro)
# header(1, The Scientific Lifecycle)
# include(Serializations.thesis_soa)
# header(1, Knowledge and Artifact Representation)
# include(Serializations.thesis_sko)
# header(1, Capturing the Scientific Lifecycle with SKOs)
# include(Serializations.thesis_app)
# header(1, Results and Conclusions)
# include(Serializations.thesis_conclusion)
# header(1, Appendix)
# include(Serializations.thesis_appendixes)
```

thesis_app

```
# header(2, A Metadata-Based Platform for Knowledge Production and Dissemination)
# include(Serializations.thesis_app_ijcai)
# header(2, A Layered Artifact Creation Environment)
# include(Serializations.thesis_app_editor)
```

thesis_appendixes

```
# header(2, LACE Thesis-Writing Details)
# include(Serializations.appendix_lace_thesis)
# header(2, SKOs-based Theory for Representing Events)
# include(Serializations.appendix_events)
# header(2, SKOs in the Classroom)
# include(Serializations.appendix_classroom)
# header(2, Promoting Scientific Inclusion)
# include(Serializations.appendix_inclusion)
# header(2, Additional Discussions)
# include(Serializations.appendix_misc)
# header(2, Footnotes)
# include(Serializations.appendix_foot)
```

appendix_foot

```
# include_footnotes(Serializations.thesis_core)
```

4. Full Semantic and Serialization Trees

node

Thesis Serialization Tree

1. Root
    1. thesis_book
        1. thesis_core
            1. thesis_app
                1. thesis_app_editor
                2. thesis_app_ijcai
            2. thesis_appendixes
                1. appendix_classroom
                2. appendix_events
                3. appendix_foot
                4. appendix_inclusion
                5. appendix_lace_thesis
                6. appendix_misc
            3. thesis_conclusion
            4. thesis_intro
            5. thesis_sko
            6. thesis_soa

node

Thesis Semantic Tree

1. Root
    1. apps
        1. apps_editor
            1. apps_editor_figure_alone_ss
            2. apps_editor_figure_content_ss
            3. apps_editor_implementation
                1. apps_editor_conclusions
                2. apps_editor_implementation_file1
                3. apps_editor_implementation_file2
                4. apps_editor_implementation_intro
                5. apps_editor_implementation_plat1
                6. apps_editor_implementation_plat2
                7. apps_editor_implementation_presentation
                8. apps_editor_implementation_semantic1
                9. apps_editor_implementation_semantic2
                10. apps_editor_implementation_semantic3
                11. apps_editor_implementation_serialization1
                12. apps_editor_implementation_serialization2
                13. apps_editor_implementation_serialization3
            4. apps_editor_intro
                1. apps_editor_semtree_ss
                2. apps_editor_sertree_ss
                3. apps_editor_services
                4. apps_editor_sfeatures_ss
                5. apps_editor_xword_ss
        2. apps_ijcai
            1. apps_ijcai_abstract

2. sko_features
    1. sko_features_credit
    2. sko_features_intproperty
    3. sko_features_intro
        1. sko_features_searchnav
            1. sko_features_searchnav_intro
            2. sko_features_searchnav_nopts
            3. sko_features_searchnav_sopts
        2. sko_features_version
            1. sko_features_version_ann1
            2. sko_features_version_ann2
            3. sko_features_version_ann3
            4. sko_features_version_ann4
            5. sko_features_version_ann5
            6. sko_features_version_intro
            7. sko_features_version_layers1
            8. sko_features_version_layers2
3. sko_file
    1. sko_file_frag1
    2. sko_file_frag2
    3. sko_file_frag3
    4. sko_file_frag_def
    5. sko_file_frag_ex
    6. sko_file_intro
    7. sko_file_node1
    8. sko_file_node2
    9. sko_file_node_def
    10. sko_file_url1
    11. sko_file_url2
    12. sko_file_url_def
4. sko_four_layers
5. sko_intro1
6. sko_intro2
7. sko_presentation
    1. sko_presentation_ann1
    2. sko_presentation_ann2
    3. sko_presentation_ann_def
    4. sko_presentation_intro
    5. sko_presentation_motivation1
    6. sko_presentation_motivation2
    7. sko_presentation_motivation3
    8. sko_presentation_node1
    9. sko_presentation_node2
    10. sko_presentation_node_def
    11. sko_presentation_purl
    12. sko_presentation_purl_ex
    13. sko_presentation_purl_rep
    14. sko_presentation_sko1
    15. sko_presentation_sko2
    16. sko_presentation_sko3
    17. sko_presentation_sko_def

2. SKOs-based Theory for Representing Events

node

node

This appendix presents a general event model to store the key metadata of an event. In particular, our model (as detailed in [60]) allows for the storage of complex subjective information relevant to the event's context. We then show how such general model can be applied to the media management issue by introducing the *experience* entity that links an event with the media representing the user's experience of such an event.

An extension of SKO specification, customized for event modeling, was used as a base of the entity representation for this work. The following are some key differences that his event SKO model has:

- *Events compose into events* : in the conventional SKO specifications nodes (content pieces) compose into SKOs (full artifacts) this differentiation does not exist when dealing with events. Events can be composed into more general events.
- *File layer is the event itself* : while the conventional SKO specification is based on digital representation of content, for events the foundational layer is the real world event itself. As such this layer is not represented in the event extension of the SKO specification.
- *Semantic layer is the metadata of the event* : the semantic layer of the conventional SKO specification is directly comparable with the metadata described in this appendix.

A follow-up paper that serializes the events thus described into news entities (used by news entities to report about the events) is currently being written.

The model is purposely kept minimal to capture the well-understood "when", "where", "what" and "who" facets of historical non-composite events; aspects defined as more interpretative dimensions are excluded. In this respect, our model differs from others since user-driven contextual metadata make events as always being subjective entities.

We believe that by clearly separating the event model from the experiential metadata of the event, this metadata is easier to use in heterogeneous applications.

1. Entities

node

The approach followed is similar to the one in [63] , where a unique entity space is proposed to store all the needed resources resources and provide a uniform representation of objects in the real and virtual world. In our model, an entity *En* is described by its metadata and associated services:

$$En = <id, type, Attr\}, \mathbf{Rel} , \mathbf{S} >$$

Where:

- *id* , is a unique identifier (e.g., an URI);
- *type* , is the type of entity, that is, the category to which it belongs to (e.g., the entity "John" is of type *Person*);
- ***Attr*** , is a set of *attributes* composed of pairs *attr = <attr_name, attr_value>* describing the properties (e.g., "date of birth") of that particular entity;
- ***Rel*** , is a set of *relational attributes* composed of pairs *rel = <rel_name, rel_value>* describing the entity's relations (e.g., "friendOf") with other entities;
- ***S*** , is a set of *services* that can be leveraged on that specific entity; for example, a service "send email" can be enabled on the *Person* entity type (etype).

An important aspect of our model is that both attributes and relationships can be further defined by *meta-attributes* . For example, attributes like "job position" or relations such as "friendOf" are provided with metadata of their own, for example to describe the time period when these are valid or the circumstances (i.e., the events) that made them true.

Another interesting aspect is the *lattice* of etypes that is encoded in the *type* property of the entity. The specific type (e.g., *Person* , *Location* , *Event*) to which the entity belongs to is used to infer its possible attributes and services. Moreover, the hierarchy defined by the lattice allows to easily define new derived etypes by just inheriting the metadata and available services of parent etypes. For example, the new etype *Author* inherits both attributes (e.g., "name", "date of birth") and services (e.g., "send email") from the etype *Person* but extends them with more specific ones (e.g., "affiliation", "get h-index").

The data structures discussed hereafter can be modeled as a set of attributes, relations and meta-attributes as formalised in the general Entity model but we provide a higher-level view for clearer reading.

2. Events

node

Events, unlike facts, are closely linked to their spatio-temporal collocation and, also, to the things constituting their subject (e.g., a sparrow in the event "a sparrow falls"). In addition, unlike objects, they have clear temporal boundaries but fuzzy spatial boundaries and they have a time-span [4] . Moreover, they are usually provided by descriptions and they may be composed of sub-events which are temporally, spatially and causally connected [64] .

In our model, we assume that "local" events are created by users and their structuring and descriptions are thus subjective.
An event entity *Ev* is modeled as follows:

Ev = <evid, t, **LEv** , Cx>

Where the elements of the tuple are:

- *evid* , the unique identifier of an event;
- *t* , the temporal collocation of an event, i.e., the time interval described by the event; it can be either a specific time interval marked off by the initial and final instants (e.g., "2009:01:14:10:00" to represent the 14th January 2009 at 10am) or a generic period of time where the temporal delimitations are not specified; also, the information on the date can be incomplete (e.g, "2009:01:::"), or relative (e.g., "the day before Christmas"), and the time interval may not be continuous as, for instance, it would happen for a "Champions League" event.
- ***LEv*** represents a set of linked events and is described in more details in the following section.
- *Cx* , the *event context* , this being regarded as a distinguishing feature of event entities. As demonstrated in [65] , [66] this context is useful for localised reasoning.
  %In principle, different contextual metadata can be provided by users describing the same "objective" event (e.g., a concert).
  The event context is represented as:

  Cx = <l, type, **Pc** >

Where:

- *l* , defines the spatial collocation of an event. It identifies a "geographical entity" such as a geopolitical entity, a natural body (e.g., mountain, river, lake), or a man-made infrastructure (e.g., building, stretch of road).
  For example, consider a person that moved around the north of Italy, the location of this

"transfer" event can be modeled by defining two geographical (point) locations (e.g., Rimini and Trento) that form the stretch of train track between Rimini and Trento.
Note that, although the spatial collocation could be objectively defined, the participant's perception of it, that is represented in the context, is itself subjective, e.g, in terms of the actual extension of the location itself ("Trento" vs. "Trentino Region").

- *type* , is the type of event (e.g., conference, trip, visit, concert);
- ***Pc*** , the event's participants is a set of relations to other entities.
  The corresponding entity values could belong to *Person* , *Organization* as well as to non-agentive etypes.
  Each relation to these participating entities is annotated with *meta-attributes* describing aspects of each entity which are only relevant in the current event's context.
  This includes the role of the entity in such an event, i.e., the modality of its participation in the event: for instance, a person can be a professor in a graduation event but is then a mother in the event describing the birthday of her daughter. Our vision of role is in line with the one given in [67] , where one of the key features of a role is that of being linked to the notion of context (the event's context in our model).

In addition, an entity participating to a given event could be described by properties valid only within the event context: for example, a temporal attribute such as "jacketColour" is only attached to the relation between a particular event and the entity.
Note that, since we regard events as subjectively perceived entities, the above mentioned properties are meant to be objective (e.g., "jacket colour") as well as subjective (e.g., "personality").
As for the attributes, relations defining the metadata of an event's participants can also be relevant only to the event's context; for instance, relations such as "girlfriendOf" or "near".

3. Events Compositionality

node

As explained before, ***LEv*** represents a set of linked events that are parts of the event to which that ***LEv*** belongs. However, to keep ***LEv*** as useful as possible for its complex event modeling purposes, the following restrictions are applied to it.

Restrictions on Time *t*

node

The time duration defined for a complex entity *CEv* , must subsume the time duration of all of its sub-events pointed by ***LEv*** . That is, if we have a function *time(Ev)* → *t*:

Given *CEv* = <*evid, t, Cx,* ***LEv*** >, □ *e* ∈***LEv*** , *time(e)* ⊑ *t*

By enforcing the previous restriction, all the individual time periods involved in the children events, are guaranteed to be subsumed in the time period of the parent complex event. This enables the representation of a complex event, like the one from our running example, in a timeline as shown in Fig. 55 .

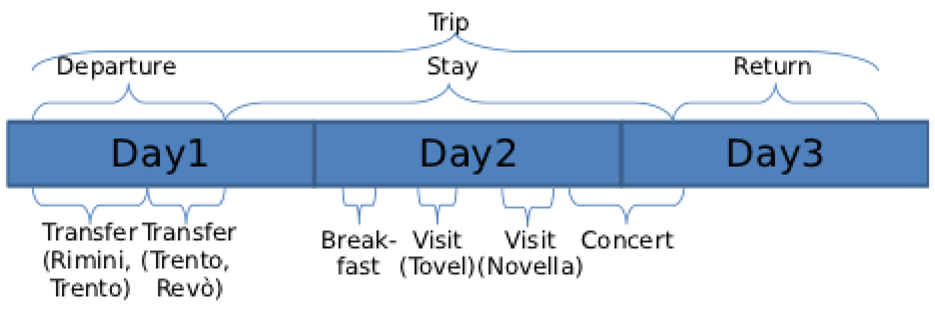


Figure 55: An example event timeline separated into tree-like sub events.

Note how in Fig. 55 all events comply with this rule. For example, the event *Stay* spans from the first to the third day and is within this period that its children events (*Breakfast* , *Visit(Tovel)* , *Visit(Novella)* and *Concert*) take place. Furthermore, note that the whole time period of *Stay* is not entirely covered by sub-events (e.g., a small period of time exists between the end of the event *Visit(Novella)* and the start of the event *Concert*). These blanks correspond to unspecified events in the trip such as, for example, the transfer between visits where she had no memorable experiences.

Restriction on Context

The context *Cx* defined for the complex entity *CEv* must subsume the context metadata of all of its sub-events pointed by ***LEv*** . That is, given the definitions of *Ev* and *Cs* , if we have the functions:

location(Cs) → l
location(Ev) → location(Cx)
participants(Cx) → **Pc**
participants(Ev) → participants(Cx)

Given CEv = <evid, t, ***LEv*** *>, Cx*
e ∈ ***LEv*** *, location(e) ⊑ location(Cx)*
□ *e ∈* ***LEv*** *, participants(e) ⊂ participants(Cx)*

By enforcing the previous restriction, all the geographic locations from the children events are guaranteed to be subsumed by the parent's location and all the participants from the sub-events are guaranteed to be included in their parent's set of participants.

From our previous examples, it is clear that the location for the event *Transfer(Rimini,Trento)* (from Fig. 55) is the railroad between the cities Rimini and Trento; furthermore, the location for the event *Transfer(Trento,Revo)* is the railroad between the cities Trento and Revo. Applying the location subsumption restriction, the location for the parent event *Departure* would be the train road between Rimini and Revo or more generally the entity for North of Italy (both of which subsume the locations from the sub-events). Likewise, the participants from *Breakfast* and *Visit(Tovel)* would also be included in the set of participants of its parent event *Stay* .

Restriction on Linked Events

node

An event cannot be included in ***LEv*** if doing so would cause the creation of a loop in the events structure. Let *connection(Start, End)* be a function that returns a sequence of events that, through their ***LEv*** components, define a directed path from the *Start* event to the *End* event (or *∅ if no such sequence exists). Then, the restriction can be expressed as:*
CEv = <evid, t, **LEv** , Cx>, connection(CEv,CEv) = ∅
This restriction is introduced to avoid the conceptual problems that would arise from an event being its own predecessor, directly or through other intermediate events.

4. Conclusions

node

The composition of events presented in this section has the following advantages:

- *Avoid repetition of information* : if there is a particular information that applies to all of the children of a complex event, instead of repeating the content on each of the sub-events, this information can be included directly at the event that is aggregating them.
  For example, thanks to this, it is not necessary to describe the weather in both the *Transfer(Rimini,Trento)* and the *Transfer(Trento,Revo)* events. If the weather did not change between these two events, the details of the weather can be included in their aggregating event *Departure* and, through compositionality, this information will apply to all of its children.
- *Capture information emerging from the aggregation* : there may exist information that emerges from the composition itself and is not part of any of its individual sub-components. For example, the whole trip could be described as being "long and tiring" but each of its sub-events may not have these properties individually. The "long and tiring" description would then only apply to the aggregation of these individual events into the *Trip* event.
- *Capture information from unspecified sub-events* : as seen in Fig. 55 , there may exist some blanks between events at high granularity levels. A lower granularity or parent event can then be used to capture information belonging to these blanks: for example, suppose we wanted to add a photo taken right after having the breakfast on the 2nd Day (*Breakfast* event) but before the visit to Tovel (*Visit(Tovel)* event); instead of adding a new event only for that photo, she could just include it in the *Stay* event.

3. SKOs in the Classroom

node

node

The exercise was based around a Mathematical Logics course, where the slides that compose the lecture can be freely annotated and contributed to by the students. All of the materials of the course (e.g. slides, references, complementary material, exercises) are stored in a repository, in such a way to support the creation of personalized versions of Mathematical Logics (or even other types of) courses that may be taught by different people in different places. The material in the repository should be organized according to its topic and kind of material and, potentially, across different versions in their evolution. The other important part of the experiment are the actual crystallized artifacts that are created (or serialized in SKO theory terms) from the repository in other to present content to other persons (in the case of our exercise, to the course's students). A diagram of this framework and of the carried out exercise is shown in Fig. 56 .

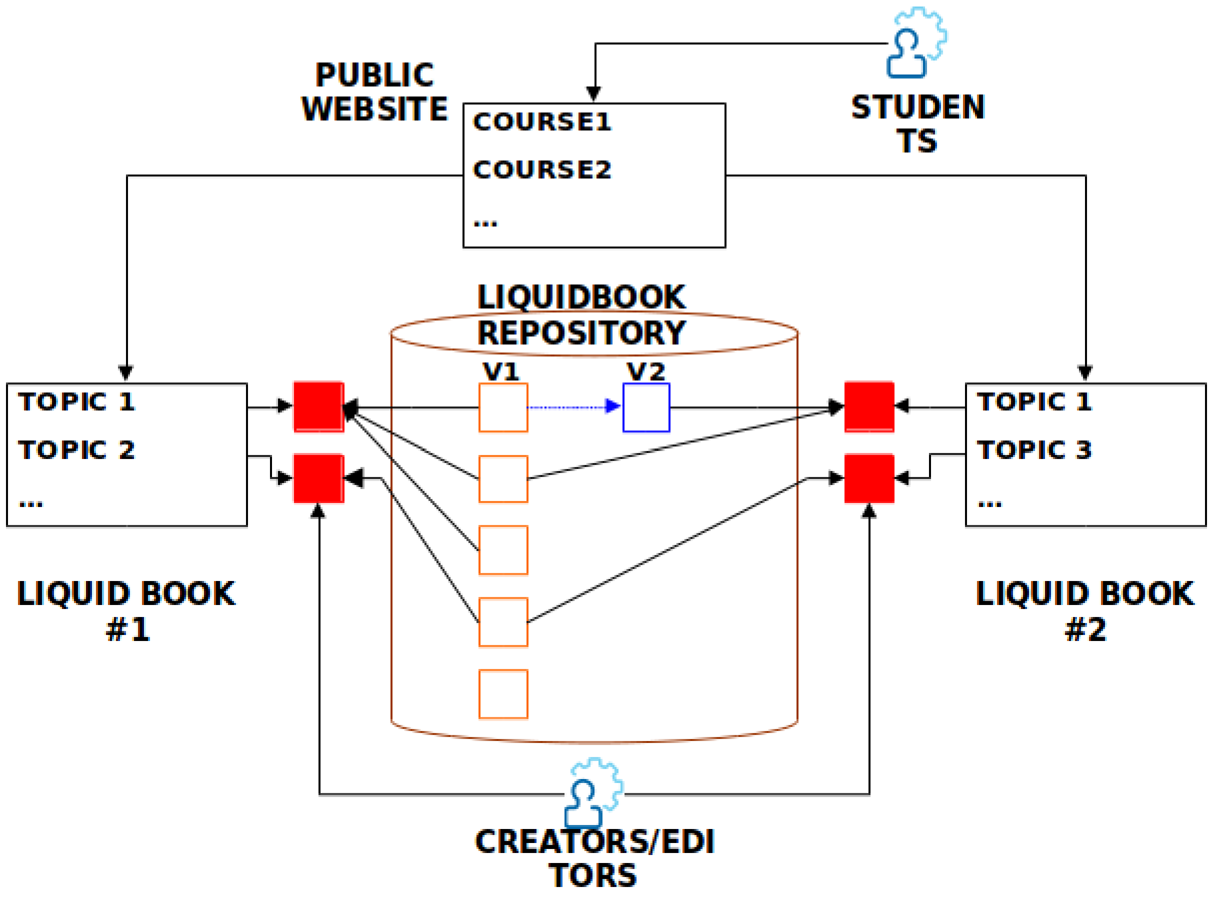


Figure 56: Different elements of the LiquidBook exercise.

The following are the key elements of Fig. 56 :

- *LiquidBook Repository* : the center of the figure is occupied by the main content repository that we will call LiquidBook repository, this is where all the resources that are considered relevant for the LiquidBook are stored [71] .
- *Resources* : the resources may be internal (like the squares drawn inside the repository), which means that the creators have the ability of modifying or producing new versions of them; or they can be external (like the red squares drawn outside the repository), which means that they may be referred to but not modified.
- *Creators/Editors* : the creators produce new versions of the Resources used in the LiquidBook Repository, while the editors add metadata and serialize the existing resources into the crystallized LiquidBooks.
- *LiquidBooks* : these represent the crystallized LiquidBooks that are serialized from the resources existing in the LiquidBook repository to present the this content to other persons in an ordered fashion. Furthermore, these LiquidBooks may even be printed and become physical solid books.
- *Public Website* : technically this is another Liquidbook (i.e. an artifact created from the resources in the repository) but in this example it has the special function of becoming the bridge for the contributions and feedback from the students.
- *Students* : normally students only consume the knowledge offered to them in a course. However, thanks to the website and the LiquidBook model, they are also able to add their submissions and corrections to the course.

1. Execution of the exercise

node

The following are the key points that were taking into consideration for carrying out the exercise:

- All the materials (e.g. presentations, exercises, references, solutions, and professor notes) are put into a specially prepared SVN repository. This SVN repository has concrete naming and placing regulations defined for files to keep it easy to access and understand.
- The lecturer gives the course lecture normally in class.
- After the lecture, the lecturer creates a sub directory in the SVN repository, where he places, the presentation and others materials that he used during his lecture. Furthermore, he can add notes and comments related to this lecture like for example "students had problems with this or that part"
- All the materials from the course are converted into webpages and published in the web page of the course. It is important to note that the original files are NOT given to the students. For example, each slide of the presentation is converted into an image file and posted independently in the course's web page. This is done mainly to encourage students to keep coming back to the course's webpage (as opposed to visiting the course webpage once to download all the slides and never coming back)
- Students are encouraged to submit their class notes, corrections, links, solutions to exercises, new exercises. These notes are directed to a previously prepared email address, where course administrators select the most insightful and interesting ones and publish at the course's webpage (directly under the slide or resource to which their refer to). A substantial incentive (5 points out of the scale) is offered to students as an incentive for submitting notes.
- For creating tests, the course manager use the course's webpage as the basic resource.
- By the end of the course, two important resources have been created:
    - The course repository: that contains the artifacts (and more generally the knowledge) from all courses.
    - The course webpage: that represents how that particular course was given.
- For future iterations of the course, the course administrators can use and evolve the course repository as they see fit. For example, the same repository could be used to teach the course in different languages and different countries (e.g. China and Paraguay), on different times (e.g. the 2010 course, the 2011 course) and different subject depth (e.g. introductory course, advanced course). As such, each instance in which a course is used will produce a new serialization of the original repository (i.e. the course's webpage) and while the repository itself does not need to be duplicated, it will evolve each time it is used to give a course.
- After several courses, the managers of the repository may decide to create a physical book on the subject matter. New materials may be created for this purpose but also some material already existing at the repository may be reused (e.g slide text, exercises and student comments/exercises from the students). Again the creation of the book would create a separate book artifact (much like the webslides from the course) and also evolve the repository of the course.

2. Preliminary Results

node

The experiment was carried out in the Mathematical Logics course [70] that started in February of 2010 and ended in June of the same year. More than 40 students were asked to send in their submissions to improve the course and, over the five months that the course lasted we compiled

over 150 submission entries from 30 different participants.
The most common type of submissions were (in no particular order):

- *Corrections to the course's slides* : this went from simple typos and pointing out corrections to some exercises, to discussing conceptual approaches and the order in which the lessons were organized.
- *Class notes* : basically, the notes that the students take during the lecture. This kind of submission is interesting for gaging what the students understand and take away from the lectures.
- *Links to external sources* : one of the most common submissions, this included links to Wikipedia articles and sometimes even papers that contained additional explanations or exercises related to the subject of the lecture.
- *Exercises and exercises with answers* : exercises related to the subject of the lecture or extending the exercises given during the lecture. Unfortunately, few of these submissions had also the answer.
- *Study Notes* : these were sent towards the end of the course and normally contained summary of concepts and formulas from the lessons

All the submissions were subjected to a review by the course owners and finally about 40 of them were selected and approved to make part of the LiquidBook repository of the course. Furthermore, a subset of these accepted submissions were published at the course's webpage; each submission placed below the slide that they actually referred to. The objective of this was to make available these corrected and approved resources to the students that had create them, in order to help them better understand the concepts of the course and prepare better for the upcoming exams.

3. Conclusions

node

This section has presented a preliminary account of the results of the first of the exercises related to a SKO-enabled LiquidBook. The reaction from the students has been positive and we believe that the information we have received through the submissions will become useful for new iterations of the Mathematical Logics course. Furthermore, this exercise also served as an early proof of concept of the File and Semantic layer of the SKO specification.

4. Promoting Scientific Inclusion

node

node

Developing countries face several challenges while trying to keep up and contribute to global scientific and technological advances. These problems persist despite the emergence of several new web-based communication opportunities that could potentially be used to bridge this knowledge gap.

The paper presented in [62] introduced a conceptual and infrastructural platform (the LiquidPub platform), based on the SKO specification for its data and metadata representation, to address these problems and opportunities.

1. Problematic

A great majority of Latin American countries are within the groups called "Scientifically Lagged Countries" and "Scientifically Developing Countries" by the World Bank [68] . According to this study, countries from these two groups are below the average in science and technology production. On the other hand; United States, Japan, and several European countries are from the "Scientifically Advanced Countries" group, which continue to account for around 90% of all research and development spending in the world. The previous fact represents also a more general concern for the governments of these countries lagging in science and technology, due to the fact that investment in research and technology is often attributed to help drive the economic growth and development of the country (e.g. Japan, India, South Korea).

Even if the science and technology are not being produced internally, it is normally in the best interest of both higher-level education institutions and companies to have access to these advances to keep themselves as updated as possible ([69]). This is especially true in highly-dynamic environments like the ones related to informatics.

Fortunately, the rapid growth of Internet has aided the communication and access to an enormous breadth of scientific and technological materials, and also bridged the physical distances existing between distant researchers. But even with the help of the network of networks, there are some challenges related to how knowledge, science and technology are handled [4] :

- Access: despite the efforts of initiatives like PLoS1 and Open Access (OAI)2, publishers are still working as "renters of access" [70] for most of the formal scientific content available on the Internet. This is specially limiting for the small institutions with small budgets that are common in developing countries.
- Search and evaluation: the massive amount and variety of information now available through the Internet may also represent a problem for resource/time-capped readers. Known as the information overload [7] , this problem makes imperative to be able to search and evaluate information at once in order to display first the results that are more relevant to the reader's needs.
- Contribution and collaboration: the current formal scientific publication systems do not favor the amounts of reuse and collaboration that would enable emerging research groups to contribute to global science.
- Discussion and consulting: participating in scientific conferences is currently one of the main methods used for discussing, making questions and networking (i.e meeting potential research collaborators or mentors). Nevertheless, budget-limited research groups have difficulties with the costs involved in attending to these conferences.

2. Contributions of a SKO-based Platform

After an overview of the of the platform architecture and the SKO specification, [62] mentions the following as possible contributions that a SKO-based platform would bring to these problems:

- Makes scientific and technical knowledge easier to find: enabling professionals, educators and struggling scientists to keep themselves competitive on their respective areas.
- Bridging distances between experts/researchers: allowing people to introduce, discuss and leave comments/annotations about scientific or technical resources regardless of their distance and origins.

- Allowing new types of collaborations and contributions: by providing a platform that it is able to keep track of authorship at the subcomponent level, thus enabling reuse and collaboration that would create new opportunities for contributing to global science. Furthermore, smaller units of contributions like data sets and experiments also become a possible contribution (as opposed of only full papers being accepted).
- Helps the dissemination of ideas: so that the work of everyone that has registered its scientific resources online is findable by the interested persons.
- Provides a common framework for scientific resources: further empowering smart search and navigation by understanding on several known formats for content and annotations, being able to link different representations of the same ideas, and different maturity states.
- Provides quick and free assessment and evaluation of published material and authors: this allows not only assessing individual contributions or authors but also groups of people, communities and, in the future, even countries. This information can be used by the researchers themselves or by the entities that regulate scientific grants, wanting to decide their investing strategies.
- Allows finding and access of different types and maturities of scientific resources: this would allow to find information in blogs or wikis (for the more innovative, albeit less stable ideas) or in journals and papers (for more mature and stable information) by using the same platform.

5. Additional Discussions

1. Metadata vs Semantic

Previous versions of the SKO theory used the word metadata to describe the second layer of the SKO specification. This name was discontinued in subsequent because based on the strict definition of metadata (i.e., data about data) all layers outside the file layer are actually metadata, as shown in Fig. 57 .

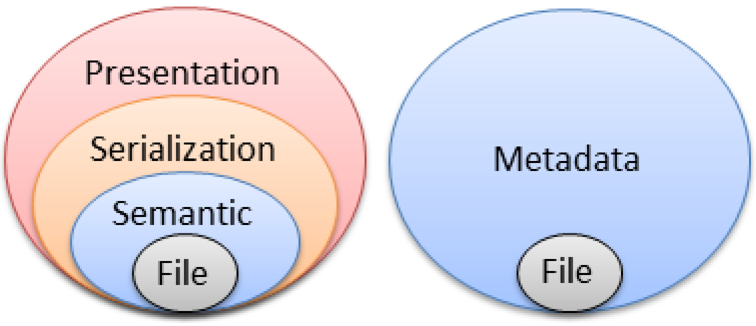


Figure 57: The second to fourth layers from the original SKO layer separation are all metadata.

Nevertheless the key insight of the SKO specification is to classify the outer metadata layers from Fig. 57 into three independent metatada sublayers (as shown in Fig. 58).

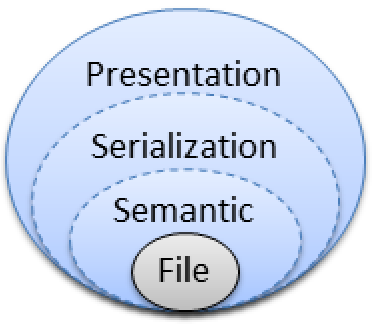


Figure 58: The SKO specification separates the metadata into three semi-independent sub-layers.

The name 'semantic' was then chosen for the 2nd layer as all the metadata in this layer (e.g., keywords, authors and other relations to other entities,) give meaning, context and place to the underlying data from the file layer.

6. Footnotes

This appendix includes all the footnotes in the thesis that appear as clickable links on the digital versions but become unreadable on printed versions.

[1] : http://project.liquidpub.org
[2] : http://www.researchgate.net
[3] : http://www.iamresearcher.com
[4] : http://www.academia.edu
[5] : http://www.informatik.uni-trier.de/~ley/db/
[6] : http://dl.acm.org
[7] : http://www.mendeley.com/
[8] : http://bazaar-vcs.org/
[9] : http://www.selenic.com/mercurial/
[10] : http://git.or.cz/
[11] : http://sourceforge.net
[12] : http://www.ohloh.net/
[13] : http://launchpad.net/
[14] : http://code.google.com/
[15] : http://en.wikipedia.org/wiki/Metadata
[16] : http://dublincore.org
[17] : http://www.foaf-project.org
[18] : http://www.openarchives.org/OAI/2.0/guidelines.htm
[19] : http://en.wikipedia.org/wiki/Hyperlinks
[20] : http://www.sec.gov/edgar.shtml
[21] : http://www.docbook.org/
[22] : http://www.w3.org/MarkUp/
[23] : http://www.w3.org/XML/
[24] : http://www.w3.org/RDF/
[25] : http://www.w3.org/TR/owl2-overview/
[26] : http://pkp.sfu.ca/lemon8

27 : http://www.facebook.com/

28 : http://www.wikipedia.com

29 : http://wiki.squeak.org/swiki/

30 : http://knol.google.com

31 : http://www.cs.univie.ac.at/project.php?pid=268

32 : http://hyp-er.wik.is/

33 : http://www.ercim.org/cyclades/

34 : http://www.galaxyzoo.org/

35 : http://cohere.open.ac.uk/

36 : http://coraal.deri.ie:8080/coraal/

37 : http://projects.kmi.open.ac.uk/scholonto/index.html

38 : http://claimaker.open.ac.uk/

39 : http://projects.kmi.open.ac.uk/scholonto/software.html

40 : http://claimblogger.open.ac.uk/

41 : http://ontoware.org/projects/swrc/

42 : http://www.aifb.uni-karlsruhe.de/english

43 : http://bibster.semanticweb.org/

44 : http://km.aifb.uni-karlsruhe.de/projects/semiport

45 : http://bibliontology.com/

46 : http://www.zotero.org/

47 : http://salt.semanticauthoring.org

48 : For a clarification on why the name 'Semantic' was chosen for this layer, instead of the name 'Metadata' that used to describe it on a previous version please check the Appendix.

49 : We are aware that is unrealistic to assume that people will separate their work by pages and not by sections or concepts. This assumption is only made here to help highlighting the specific concepts being explained but note that the concepts also hold for more realistic examples.

51 : Note that in this figure the arrows actually represent actions and not relations between the entities as it was previously the case in the document.

52 : http://www.mpi-inf.mpg.de/yago-naga/yago/

55 : http://scholar.google.com

56 : http://www.eswc2011.org/

57 : http://www.jcdl.org/

58 : http://www.easychair.org/

53 : http://www.freebase.com/

54 : http://ijcai.org

59 : http://www.executablepapers.com/

60 : http://getutopia.com/documents/

61 : http://collage.cyfronet.pl/

62 : http://collage.cyfronet.pl/

63 : http://en.wikipedia.org/wiki/Mindmaps

64 : http://www.xwiki.org/

65 : http://velocity.apache.org/engine/releases/velocity-1.5/user-guide.html

66 : http://groovy.codehaus.org/

67 : At least for a short time following the defense of this thesis.

68 : http://www.w3.org/Style/CSS/

48 : For a clarification on why the name 'Semantic' was chosen for this layer, instead of the name 'Metadata' that used to describe it on a previous version please check the Appendix.

70 : http://disi.unitn.it/~ldkr/ml2010/

71 : For simplicity, we will assume that this central repository indeed exists; nevertheless,according to the SKO structural and LiquidBook definitions, this repository may not be centralized or even it may even be the "Internet as a whole". This means, as previously implied in the SKO definition,that

keeping the url and metadata each of the used resources should be enough to manage them and used them in LiquidBooks.

node

Bibliography

[1] Latour, B.: Science in Action: How to Follow Scientists and Engineers through. Harvard University Press, Cambridge Mass., USA (1987).

[2] Kornfeld, W. A., & Hewitt, C.: The Scientific Community Metaphor. IEEE Transactions on Systems, Man, and Cybernetics , 11 (1981).

[3] European Commission: Study on the Economic and Technical Evolution of the Scientific Publication (2006).

[4] Casati, F., Giunchiglia, F., Marchese, M.: Publish and perish: why the current publication and review model is killing research and wasting your money. ACM Ubiquity (November 2006).

[5] Parnas, D. L.: Stop the Numbers Game. Communications of the ACM , 50 (2007).

[6] Rothwell, P., & M., C.: Reproducibility of peer review in clinical neuroscience. Brain 123 (2000).

[7] Edmunds, A., & Morris, A.: "The problem of information overload in business organisations: a review of the literature". International Journal of Information Management , 17-28 (2000)

[8] Beck, K.: Embracing change with extreme programming. , vol. 32, no. 10. 1999. Computer 32, 10 (1999).

[9] Asklund, U., Bendix, L.: The Unified Extensional Versioning Model, Lecture Notes in Computer Science Volume 1675. Springer, 1999.

[10] Fraser, C. W., Myers, E. W.: An editor for revision control. ACM Transactions on Programming Languages and Systems 9, 2 (1987).

[11] Raya, I., ZHANGB, J.: Towards a new standard for allowing concurrency and ensuring consistency in revision control systems. Computer Standards and Interfaces 29, 3 (2007).

[12] Fogel, K.: Producing open source software: How to run a successful free software project (2007).

[13] Christley, S., Madey, G.: Collection of activity data for sourceforge projects. Tech. Rep. TR-2005-15, University of Notre Dame (2005).

[14] Baader, F., McGuinness, D.L., Nardi, D.: The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press (2003).

[15] Bush, V.: As we may think. Atlantic Monthly 176, 101-108 (1945).

[16] Gemmell, J., Bell, G., Lueder, R., Drucker, S., Wong, C. Mylifebits: Fulfilling the memex vision. In Proceedings of ACM Multimedia, pp. 235-238 (2002).

[17] Dourish P., Edwards K., LaMarca A., Lamping J., Petersen K., Salisbury M., Terry, D., Thornton J.: Extending document management systems with user-specific active properties. ACM Trans Info Sys 18, 2, 140-170 (2000).

[18] LaMarca , A., Edwards, K., Dourish, P., Lamping, J., Smith, I., Thornton, J.: Taking the work out of workflow: Mechanisms for document-centric collaboration. In Proceedings of the 6th European Conference on Computer-Supported Cooperative Work (ECSCW '99, Copenhagen, Denmark, Sept. 12-16), Kluwer Academic, Dordrecht, Netherlands (1999).

[19] Wang, J., Kumar, A.: A framework for document-driven workflow systems. Springer, Berlin, pp. 285-301 (2005).

[20] Nigam, A., Caswell, N. S.: Business artifacts: An approach to operational specification. IBM Systems Journal 42, 3, 428-445 (2003).

[21] Wellman, B., Salaff, J.: Computer networks as social networks: Collaborative work, telework, and virtual community. Annual Review of Sociology 22 (1996).

[22] Kuma, R., Novak, J., Tomkins, A.: Structure and evolution of online social networks. In International Conference on Knowledge Discovery and Data Mining (2006).

[23] Grudin, J.: Computer-supported cooperative work: Its history and participation. Computer 27, 4, 19-26 (1994).

[24] Carstensen, P.H., Schmidt, K.: Computer supported cooperative work: new challenges to systems design, 1999. At http://citeseer.ist.psu.edu/carstensen99computer.html.

[25] Nardi, B.A., Whittaker, S., Bradner, E.: Interaction and outeraction: instant messaging in action. In Proceedings of the 2000 ACM conference on Computer supported cooperative work, ACM Press New York, NY, USA, pp. 79-88 (2000).

[26] Dourish, P., Bellotti, V.: Awareness and coordination in shared workspaces. In ACM conference on Computer-supported cooperative work, pp. 107-114 (1992).

[27] Greenberg, S., Marwood, D.: Real time groupware as a distributed system: concurrency control and its effect on the interface. In ACM conference on Computer supported cooperative work, pp. 207-217 (1994).

[28] Jacovi, M., Soroka, V.: The chasms of cscw: a citation graph analysis of the cscw conference. In Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work (2006).

[29] Naak, A., Hage, H., Aimeur, E.: Papyres: A Research Paper Management System. E-Commerce Technology and Enterprise Computing, E-Commerce and E-Services, IEEE Conference and Fifth IEEE Conference (2008).

[50] Groza, T., Handschuh, S., Clark, T., Buckingham Shum, S. and de Waard, A.: A short survey of discourse representation models. In: Proceedings 8th International Semantic Web Conference, Workshop on Semantic Web Applications in Scientific Discourse. Lecture Notes in Computer Science, Springer Verlag: Berlin, Washington DC (2009).

[31] De Waard, A., Tel, G.: The ABCDE format - enabling semantic conference proceeding. In: Proceedings of 1st Workshop: "SemWiki2006 - From Wiki to Semantics", Budva, Montenegro. (2006)

[32] Buckingham Shum, S., Motta, E., & Domingue, J.: Modelling and Visualizing Perspectives in Internet Digital Libraries. Proceedings of ECDL'99: Third European Conference on Research and Advanced Technology for Digital Libraries (1999).

[33] Uren, V., Buckingham Shum, S., Li, G., & Bachler, M.: "Sensemaking Tools for Understanding Research Literatures: Design, Implementation and User Evaluation". Human Computer Studies , 420-445 (2006).

[34] Sereno, B., Shum, S. B., Motta, E.: Formalization, user strategy and interaction design: Users' behaviour with discourse tagging semantics. In Proceedings Workshop on Social and Collaborative Construction of Structured Knowledge, 16th International World Wide Web Conference (2007).

[35] Sure, Y., Bloehdorn, S., Haase, P., Hartmann, J. and Oberle., D.: The SWRC ontology - semantic web for research communities. In EPIA, pages 218-231, (2005).

[36] Groza, T., Handschuh, S., Möller, K., Decker, S.: SALT - Semantically Annotated LaTeX for Scientific Publications. Lecture Notes in Computer Science. Volume 4519/2007, pages 518-532 (2007).

[37] Fabio Casati, Fausto Giunchiglia , Maurizio Marchese. Liquid publications: Scientific publications meet the web. Tech Report from Unitn, found at http://eprints.biblio.unitn.it/archive/00001313/01/073.pdf , 09 2007.

[38] Grosso, P., Maler, E., Marsh, J., and Walsh, N.: Xpointer element() scheme, 2003.

[39] Thao, C., Munson, E. V., & Nguyen, T. N.: Software Configuration Management for Product Derivation in Software Product Families. ECBS '08: Proceedings of the 15th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems (2008)

[40] Zaihrayeu, I., Sun, L., Giunchiglia, F., Pan, W., Ju, Q., Chi, M., Huang, X.: From Web Directories to Ontologies: Natural Language. Processing Challenges. ISWC/ASWC: 623-636, (2007).

[41] Giunchiglia, F., Zaihrayeu, I.: Lightweight Ontologies. Encyclopedia of Database Systems: 1613-1619, (2009).

[42] Giunchiglia, F., Biswanath, D., and Vincenzo, M.: Faceted Lightweight Ontologies, Conceptual Modeling. Foundations and Applications Lecture Notes in Computer Science, Volume 5600/2009, 36-51, DOI: 10.1007/978-3-642-02463-4 3 (2009).

[43] Giunchiglia, F., Marchese, M., Zaihrayeu, I: Encoding Classifications into Lightweight Ontologies. J. Data Semantics vol 8: 57-81, (2007).

[44] Autayeu, A., Giunchiglia, F., Andrews, P.: Lightweight Parsing of Classifications into Lightweight Ontologies Research and Advanced Technology for Digital Libraries, 14th European Conference, ECDL2010, Glasgow, UK (2010)

[45] Giunchiglia, F., Yatskevich, M., Avesani, P., Shvaiko, P.: A Large Scale Dataset for the Evaluation of Ontology Matching Systems. The Knowledge Engineering Review Journal (KER), Cambridge University Press, issue 24, number 2 (2009).

[46] Giunchiglia, F., Kharkevich, U., Zaihrayeu, I., "Concept search". In Proceedings of ESWC, (2009).

[47] Giunchiglia, F., Shvaiko, P., Yatskevich, M., S-Match: an Algorithm and an Implementation of Semantic Matching, in proceedings First European Semantic Web Symposium, ESWS (2004).

[48] Giunchiglia, F., and Autayeu, A., and Pane, J.: "S-Match: an open source framework for matching lightweight ontologies". IOS Press, Semantic Web Journal, May 31 (2011).

[49] Groth, P., Gibson2, A., Velterop, J.: The anatomy of a nanopublication. Information Services and Use. IOS Press. Volume 30, Number 1-2, 51-56 (2010).

[50] Groth, P., Gibson2, A., Velterop, J.: The anatomy of a nanopublication. Information Services and Use. IOS Press. Volume 30, Number 1-2, 51-56 (2010).

[51] Groth, P., Gibson2, A., Velterop, J.: The anatomy of a nanopublication. Information Services and Use. IOS Press. Volume 30, Number 1-2, 51-56 (2010).

[52] Kunze, J., Hu, R., Cruse, T., Mitchell, C., Abrams, S., Hastings, K., Schiff, L.: Baby steps to data publication. California Digital Library. 9 December (2010).

[53] Eppler, M. J.: A Comparison between Concept Maps, Mind Maps, Conceptual Diagrams, and Visual Metaphors as Complementary Tools for Knowledge Construction and Sharing. Information Visualization September 21, vol. 5 no. 3 202-21 (2006).

[54] Ivanyukovich, A., Marchese, M.: Unsupervised Metadata Extraction in Scientific Digital Libraries Using A-Priori Domain-Specific Knowledge. SWAP 2006 - Semantic Web Applications and Perspectives, Proceedings of the 3rd Italian Semantic Web Workshop, Scuola Normale Superiore, Pisa, Italy, 18-20 December (2006).

[55] F. Giunchiglia, H. Xu, A. Birukou, R. Chenu-Abente. Scientific Knowledge Object Patterns. Proceedings of the 15th European Conference on Pattern Languages of Programs (EuroPLoP), Irsee Monastery, Bavaria, Germany, July 7 - July 11 (2010).

[56] M. Baez, A. Birukou, R. Chenu-Abente, M. Medo, N. Osman, D. Ponte, J. Sabater-mir, L. Schneider, M. Turrini, G. Veltri, J.R. Wakeling, H. Xu. State of the Art in Scientific Knowledge Creation, Dissemination, Evaluation and Maintenance. Technical Report DISI-09-067, Ingegneria e Scienza dell'Informazione, University of Trento. 2009.

[57] R. Chenu-Abente, F. Giunchiglia, L. Cernuzzi. From Software to Artifacts: Supporting the Current Scientific Knowledge Needs. Congreso Iberoamericano en "Software Engineering" (CIbSE), Rio de Janeiro, Brasil, 27 April - 29 April, 2011.

[58] F. Giunchiglia, R. Chenu-Abente. Scientific Knowledge Objects v.1. Technical Report DISI-09-006, Ingegneria e Scienza dell'Informazione, University of Trento. 2009.

[59] F. Giunchiglia, R. Chenu-Abente. Papers, Persons and Events: an Integrated Entity-Based Platform for Knowledge Production and Dissemination. Festschrift in honor of Ramon López de Màntaras. Consejo Superior de Investigaciones Scientificas - Instituto de Investigación en Inteligencia Artificial (CSIC - IIIA), 2012.

[60] F. Giunchiglia, P. Andrews, G. Trecarichi, R. Chenu-Abente. Media Aggregation via Events. First International Workshop "EVENTS 2010 - Recognizing and tracking events on the Web and in real life". The 6th Hellenic Conference on Artificial Intelligence (SETN 2010), Athens, Greece. May 2010.

[61] F. Giunchiglia, H. Xu, A. Birukou, R. Chenu-Abente. Scientific Knowledge Object Patterns. Proceedings of the 15th European Conference on Pattern Languages of Programs (EuroPLoP), Irsee Monastery, Bavaria, Germany, July 7 - July 11, 2010.
Chenu-Abente, Ronald and Jara, Juan Jose and Giunchiglia, Fausto and Casati, Fabio and Marchese, Maurizio (2010) Supporting Scientific

[62] R. Chenu-Abente, J.J. Jara, F. Giunchiglia, F. Casati, M. Marchese: "Supporting Scientific Inclusion: Novel Approaches for the Representation and Assessment of Scientific Knowledge Objects". XXXVI Conferencia Latinoamericana de Informática (CLEI), Asunción, Paraguay. October 18 - October 22, 2010.

[63] Bouquet, P., Stoermer, H., Niederee, C., Mana, A.: Entity name system: The backbone of an open and scalable web of data. In: In Proc. of IEEE-ICSC. 554-561 (2008).

[64] Scheffler, U.: Events as shadowy entities. Konstanzer Berichte: Logik und Wissenschaftstheorie 28 (1992).

[65] Giunchiglia, F.: Contextual reasoning. Epistemologia 16(I Linguaggi e le Macchine), 345-364 (1993).

[66] Ghidini, C., Giunchiglia, F.: Local Models Semantics, or Contextual Reasoning Locality + Compatibility. Artificial Intelligence 127, 221-259 (2001).

[67] Masolo, C., Vieu, L., Bottazzi, E., Catenacci, C., Ferrario, R., Gangemi, A., Guarino, N.: Social roles and their descriptions. In: Proc. of Knowledge Representation Workshop, AAAI Press, 267-277 (2004).

[68] Wagner, C., Brahmakulam, I., Jackson, B. et al.: "Science and Technology Collaboration: Building Capacity in Developing Countries?". Report prepared for the World Bank by RAND Science and Technology (2001).

[69] Harris, E.: Building scientific capacity in developing countries. Pubmed Central, EMBO Rep.; 5(1): 7-11 (January 2004)

[70] Cuel, R., Ponte, D., Rossi, A.: "Towards an Open/Web 2.0 Scientific Publishing Industry? Preliminary Findings and Open Issues". http://wiki.liquidpub.org/mediawiki/upload/b/b3/CuelPonteRossi09.pdf (2009).