# Table of Contents

## I  Keynote Abstracts

<table>
<thead>
<tr>
<th>Title</th>
<th>Author(s)</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Incubação do Projecto Cloud@RCTS</td>
<td>Pedro Assis</td>
<td>3</td>
</tr>
<tr>
<td>Opening A Whole New World with HTML5</td>
<td>Tiago Andrade e Silva</td>
<td>19</td>
</tr>
</tbody>
</table>

## II  Full Papers

<table>
<thead>
<tr>
<th>Title</th>
<th>Author(s)</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>An XMPP messaging infrastructure for a mobile held security identity wallet of personal and private dynamic identity attributes</td>
<td>Alexandre Augusto &amp; Manuel Correia</td>
<td>23</td>
</tr>
<tr>
<td>PExIL: Programming exercises interoperability language</td>
<td>Ricardo Queirós &amp; José Paulo Leal</td>
<td>37</td>
</tr>
<tr>
<td>Using mobile device detection approaches to augment the accuracy of web delivery content</td>
<td>Ricardo Queirós &amp; Mário Pinto</td>
<td>49</td>
</tr>
<tr>
<td>XML schemas for parallel corpora</td>
<td>Alberto Simões &amp; Sara Fernandes</td>
<td>59</td>
</tr>
</tbody>
</table>
XCentric-based visual approach to web content verification
* Liliana Alexandre & Jorge Coelho ..................... 71

Comparing application profiles and ontologies for describing experiment data
* João Silva, Cristina Ribeiro & João Correia Lopes 83

An engine for generating XSLT from examples
* José Paulo Leal & Ricardo Queirós ....................... 91

xml2pm: A tool for automatic creation of object definitions based on XML instances
* Nuno Carvalho, Alberto Simões & J.João Almeida 103

DWXML: A preservation format for data warehouses
* Carlos Aldeias, Gabriel David & Cristina Ribeiro . 115

Using ontologies in database preservation
* Ricardo Freitas & José Carlos Ramalho ............... 127

Author Index .......................................................... 138
Editorial

These are the proceedings for the ninth national conference on XML, its Associated Technologies and its Applications (XATA’2011). It is the first time that XATA is organized in two consecutive years at the same place with the same organizers.

Last year we tried to change the tendency of previous years, where interest in the conference has declined, and we were able to duplicate the number of papers presented in the previous edition (2009, when XATA was held together with Infórum).

Unfortunately the interest in the conference declined again. For the first time the organizers issued a third call for papers, and only twelve proposals were received. From these, 10 were accepted as full papers for presentation.

Nevertheless, this XATA edition added some attractive news that were expected to foster the interest in the conference: the best papers’ authors will be invited to submit extended versions of their papers to a book publication by IGI Global. Also, it is the first edition that has the two main companies developing tools for XML as sponsors.

As a participation incentive, this XATA edition includes two keynotes, on different subjects: XML as the communication mean for Cloud Computing, and HTML5, the new standard for the web. It will also include a Pecha Kucha session, where authors will be challenged to present talks of 20 slides, each taking 20 seconds.

Finally, we would like to thank all authors for their work and interest in the conference, and to the scientific committee members for their review work.

Alberto Simões

VII
Steering Committee

Cristina Ribeiro (FEUP and INESC Porto)
Gabriel David (FEUP and INESC Porto)
João Correia Lopes (FEUP and INESC Porto)
José Carlos Ramalho (DI/UM)
José Paulo Leal (FCUP)
Pedro Henriques (DI/UM)

Organizing Committee

Alberto Simões (CCTC/UM)
Luís Correia (ESEIG/IPP)
Mário Pinto (ESEIG/IPP)
Ricardo Queirós (ESEIG/IPP)
Scientific Committee

Ademar Aguiar (FEUP and INESC Porto)
Alberto Simões (ESEIG/IPP)
Alda Lopes Gangarski (Institut N. des Télécommunications)
Ana Paula Afonso (DI/FC/UL)
Benedita Malheiro (ISEP)
Cristina Ribeiro (FEUP and INESC Porto)
Daniela da Cruz (DI/UM)
Francisco Couto (DI/FC/UL)
Gabriel David (FEUP and INESC Porto)
Giovani Librelotto (UFSM)
João Correia Lopes (FEUP and INESC Porto)
João Moura Pires (FCT/UNL)
José Carlos Ramalho (DI/UM)
José João Almeida (DI/UM)
José Paulo Leal (DCC/FCUP)
Luis Ferreira (EST/IPCA)
Marta Jacinto (ITIJ)
Miguel Ferreira (DSI/UM)
Nuno Carvalho (DI/UM)
Nuno Oliveira (DI/UM)
Paulo Marques (DEI/UC)
Pedro Rangel Henriques (DI/UM)
Ricardo Queirós (ESEIG/IPP)
Rui Castro (KEEP)
Rui Lopes (DI/FC/UL)
Salvador Abreu (DI/UE)
Stephane Gançarski (LIP6, University P. & M. Curie)
Xavier Gómez Guinovart (UVigo)
Keynote Abstracts
Federation of Academic Clouds: 
The Next Step in Higher Education?

Pedro Assis
Engineering Institute of the Porto Polytechnic
pfa@isep.ipp.pt

Abstract. Cloud computing paradigm presents as a feasible solution to address the Bologna Process 2020 challenge “at least 20% of those graduating in the European higher education area should have had a study or training period abroad.” According to the author, in the near-future, academic clouds will involve European higher education institutions that, eventually, had no prior contact with cloud computing technology, and with different backgrounds. It is argued that the Cloud@RTS proposal plus current efforts being made by the scientific community, in what concerns security, privacy, reliability, interoperability and management, will allow such multidisciplinary cooperation among higher education partners. Hopefully, this will brings together institutions, build stronger bonds and promote the Bologna mobility and cooperation spirit.

Keynotes: Bologna Process, European Higher Education Area, Globalization, Cloud Computing, Web Technologies, Standardization, Cloud@RTS

1 Introduction

The 29 signatories of the Bologna Declaration sought to “create a European Higher Education Area (EHEA) of international cooperation and academic exchange attractive to European students” inclusive to non-Europeans. Currently, 47 countries commit to “facilitate mobility of students, graduates and higher education staff” among higher education institutions (HEIs) despite cultural, administrative and scientific barriers that political agreement could not override.

To overcome these barriers Géant (www.geant.net) projects eduroam and eduGAIN support “physical mobility” as their focus is on world-wide roaming Internet access service and a confederation of identity providers, respectively. Although such initiatives contribute to Bologna fulfillment they are still not enough. It is the author’s conviction that, from a technical point of view, to achieve a true Bologna “mobility space” it is needed a platform that supports “information mobility” in a seamlessly and secure way regardless of user provenance. It is this keynote proposal that such platform will be based on the emerging Cloud computing paradigm. Cloud computing is both a deployment and a service model that aims Information Communication Technologies (ICT) platforms transformation in such a way that they will become elastic, highly-available, fault-tolerant, secure and multi-tenant. The deployment of cloud ecosystems shall provide ubiquity access to both applications and data regardless user location.
In the near-future, academic clouds will involve European HEIs that, eventually, had no prior contact with Cloud computing technology, and have different backgrounds. Such multidisciplinary cooperation among partners is one of the proposal strong points: The input of participating institutions does not originate in one specific area but it rather comprises different areas, bringing together institutions that had not had joint projects and promoting cooperation among different areas. This scenario opens a whole new trend of international cooperation, build stronger bonds among European HEIs and promote the implementation of Bologna mobility and cooperation spirit.

This keynote provides an analysis regarding the usage of a hybrid model federated space of academic clouds as a feasible solution to promote mobility, addressing cloud heterogeneity issues while endorsing their interoperability. Such scenario will leverage current eduroam and eduGAIN services allowing innovation on ICT-based content, services, pedagogics, lifelong-learning and mobility support, increasing collaborative work, information availability, reducing information transmission stages, thus increasing information reliability, reducing information costs of and increasing student-teacher proximity. This is e-learning and e-teaching taken forward to the next decade paradigm.

2 The Bologna Process (Un)Success!

The premiss of a Europe of knowledge as “an irreplaceable factor for social and human growth and as an indispensable component to consolidate and enrich the European citizenship” (Bologna Declaration 1999) drove the deployment of the European higher education area. Officially lunched on 11-12 March 2010, EHEA addresses the need for international harmonization in academic degrees, enabling cross-credits processes and international multi-awards. Ultimately, EHEA reflects the need to think Europe’s higher education on a global scale, improving its competitiveness, internationalisation and mobility. Such objectives are described in ten action lines, which reflect key impediments, identified by the signatories, that must be overcome: the adoption of a system of easily readable and comparable degrees; system of credits definition; promotion of mobility; lifelong learning, and rise European education standards to seize non-European, highly quality and motivated students, researchers and teachers.

Although Bologna Process legal framework has been ratified by 45 of its members, such recognition by itself will not ensure successful practices. As Bergan [7] reported “Practice, however, is less promising. A key principle of the Council of Europe/UNESCO Convention is that foreign qualifications should be recognized unless you can demonstrate that there is a substantial difference between the qualification for which recognition is sought and the corresponding qualification of the home country. To say that this requires more attention is an understatement.” European HEIs must endorse EHEA spirit and actively take part of this process. Higher education institutions must go beyond harmonization in academic degrees and cross-credits processes, and commit to true student mobility regardless of minor curricula misalignments, economical obstacles, organizational and cultural differences. Such imperfections must be dealt with, eventually accepted, but most important, they must be put in perspective: by no means they can compromise Bologna fulfillment.
European Commission (EC) has been playing an important role in public support to Bologna Process: Action programs Erasmus Mundus and Tempus. Through these projects the EC gained significant influence over Bologna destinies, albeit formally the EC has no competence over this process. Nonetheless, is a relevant support to student and staff mobility and HEIs internationalisation. According to IDP Education Australia study [10] concerning the global demand for international education, at the beginning of the 21st century there were an estimated 1.8 million students worldwide in mobility. By the year 2025 the same study projects an increase to 7.2 million, being China and India the two major contributors for this evolution. Although the global financial and economic crisis is putting under great pressure Europeans HEIs’ main source of funding (public), Europe cannot afford to be lagged behind while others, like Australia with a projected value of one million foreign students by 2025, take the lead benefiting from geopolitical assessment and appealing higher education systems. In the future, to achieve sustainability, HEIs should reduce costs through the integration of education and research resources; globalize their activities widening partnerships among HEIs and increase their mutual portfolio; promote lifelong learning and support non-traditional learning paths; and promote mobility and exchange programs to enhance student and staff quality and strengthen globalization. Such hindrance to HEIs development should be taken as an opportunity to innovate. In this context, it is propose the use of Cloud computing paradigm to deploy HEIs administrative, academic and research Information and Communication Technologies (ICT) platforms to achieve information sharing transparency, reducing the stages for information transmission, thus increasing reliability, and above all promote collaborative work.

3 Cloud Computing

In 2008 a joint workshop between Educause Campus Cyberinfrastructure (CCI) Working Group and the Coalition for Academic Scientific Computation (CASC), both North American agencies, addressed the growth of Information Technologies (IT) in research and education over the last decade. In the end, existing cyberinfrastructures (CI) were characterized, as well as identified the main hurdles in the path of development and deployment a coherent CI ecosystem. The attendees recognized that common cyberinfrastructures lack integration and interoperability, resulting in significant operational costs. The meeting outcome established the following key principles: “Harnessing campus and national resources”; “Information life cycle: accessibility, usability, and sustainability”; “Identity management, authentication, and authorization”; and “Human resources and broader impact” (Educause CCI 2009). From the subsequent debate several strategic recommendations were layout, including: Join ventures among HEIs, resource providers and governmental agencies to support, promote and develop a coherent, comprehensive set of computing and data facilities; contribute for a sustainable development based on standards for data provenance, metadata, discoverability, and openness; deploy

1 For purpose of the cited report, cyberinfrastructure comprehends “computational systems, data and information management, advanced instruments, visualization environments, and people, all linked together by software and advanced networks to improve scholarly productivity and enable knowledge breakthroughs and discoveries not otherwise possible.” [20].
common identity management, authentication, and authorization procedures, likewise Géant projects eduroam and eduGAIN; and develop technologies and tools to use the emerging CI for education and scholarship. To a great extend Cloud computing own requirements and aims are in fact align with the analysis made for CI infrastructures. In face of such similarities, it only natural to argue that a set of interoperable CI ecosystems should have similar characteristics than a hybrid model federated space of academic clouds.

Cloud computing paradigm has been under scrutiny of researchers and business. In-between criticisms and praises, Cloud computing is affirming itself as being capable of integrate existing technologies and tools in its ecosystem. In the author’s view, the technologies and standards reuse, on demand provisioning (elasticity) and new business model (pay-as-you-go), are among Cloud computing highlights that justify this paradigm added value for ICT evolution.

The roots of Cloud computing lay in utility computing back in the 1990s, as Application Service Providers (ASP) started to deliver software as a service. Web services followed, and with them the promise of a new model for software delivery based on a registry that supports dynamic binding and discovery. Tightly couple with Web services, Service-Oriented Architecture (SOA) generalized the service provider-consumer pattern. Finally, Grid computing stands side-by-side with Cloud computing, although the latest offers much more than a simple batch submission interface. According to Kealey et al. (2009), “Cloud computing represents a fundamental change from the Grid computing assumption: when a remote user “leases” a resource, the service provider turns control of that resource over to the user.”

3.1 What, Why and How

According to Mell and Grace, National Institute of Standards and Technology (NIST) researchers, Cloud computing is both a deployment and service model [26]. Such paradigm aims to transform ICT platforms in such a way that they will become elastic, highly available, fault tolerant, secure and multi-tenant. As this transformation takes place, it is expected that ICT technicians will focus their work on their companies’ core business, and not on technology complexity. Such complexity has been referred by others, e.g., Kephart and Chess, as the “main obstacle to further progress in the ICT industry” [24]. According to these IBM researchers, the solution, and challenge, is to develop self-managing computing systems. This vision, named Autonomic Computing, is related with natural self-organizing systems, which account for large numbers of interacting components at different levels. According to autonomic computing community, the way such systems work offer inspiration as how to tackle ICT complexity.

OpenCrowd (www.opencrowd.com) taxonomy addresses four areas: Infrastructure services (e.g. storage, and computational resources), Cloud services (e.g. appliances, file storage and cloud management), Platform services (e.g. business intelligence, database, and development and testing); and Software services (e.g. billing, financial, legal, sales, desktop productivity, human resources, and content management). On the other hand, NIST advises that Cloud computing should offer three main types of services, each addressing specific user needs: Infrastructure as a Service (IaaS), Platform as a
Service (PaaS) and Software as a Service (SaaS). IaaS offers the provision of raw computing resources including processing, storage, and network. The consumer has control over the assigned resources, not over the underlying cloud platform. Among the examples of Cloud computing Frameworks (CCF) are OpenNebula, Eucalyptus and Nimbus. PaaS provides a development platform, comprising programming languages and tools, that enables consumer to develop and deploy applications onto the cloud infrastructure. Following the e-Science initiative (www.eu-egee.org), European higher education institutions currently providing Grid Computing services would be integrated in this virtual infra-structure, offering Grid services as platform as a service. This way, it would possible the cohabitation of native Grid applications as resources in a Cloud computing Ecosystem, likewise Google Apps, Microsoft Windows Azure, SalesForce.com and others. Finally, SaaS provisions applications/services running on top of the cloud platform. Consumer doesn’t have any control, but over user configuration data (e.g. Facebook, Gmail). The main difference between these two taxonomies is due to the emphasis that OpenCrowd places on the need to “create customize clouds,” while Mell and Grace work does not.

Cloud services are made available through different deployment models. Mell and Grace envisage the followings: Private Cloud, it is operated by a single entity; Community Cloud, it is operated by a set of organizations that share common interests; Public Cloud, it is made available to the public or large industry group. It is owned by an organization that sells cloud services; Hybrid Cloud, compositions of two or more clouds as described before. Such organization models do not require in-house cloud infrastructure, neither its management and control. This can be provided by a third party under an outsourcing agreement.

In the real world, Cloud computing should provide the means to handle user demand for services, applications, data, and infrastructure in such a way that these requests can be rapidly orchestrated, provisioned, and scale up/down through a pool of resources related with computing, networking, and storing facilities.

### 3.2 Challenges

Cloud computing is emerging in a mist of concerns and challenges. A recent Computer Associates survey [11] lists security, privacy, reliability, interoperability and management among the main concerns of European IT managers regarding the adoption of Cloud computing. To cope with security issues, cloud providers and researchers are embedding encryption technologies [2] to protect information before storing it on the cloud. Also, cloud services are advertising the support of secure transactions (Amazon VPC [3]) between client premises and the cloud, enabling the usage of common management and control policies for both physical (local) and virtual (remote) resources. Recently (2008) Cloud Security Alliance (CSA) was announced to promote the use of best practices for providing security assurance within Cloud computing, and offer education on the uses of Cloud computing to help secure all other forms of computing [14].

In what concerns privacy, Cloud computing providers must obey to the existing data protection laws that enforce privacy regulations in each nation. The Portuguese
Comissão Nacional de Protecção de Dados and in UK’s Data Protection Act are two examples of independent bodies that are endowed with the power to supervise and monitor compliance with the Portuguese and British laws in the area of personal data protection. One particular issue that major cloud providers are already addressing is the prohibition, for some European countries, to transmit and store personal records outside the European Union (EU). To address this issue, Google and Amazon are deploying storage facilities inside EU space. Several outages were reported (years 2008 and 2009) by Microsoft, Google, SalesForce and Amazon in different types of services. Although such breakdowns are uncommon and according to service providers all data was restored, the lack of service and eventual loss of data can cause much damage, specially to large scale companies. On the contrary, small to medium enterprises can, eventually, be more tolerant to such faults, as management procedures and technical support follow less stringent requirements. Nonetheless, these events are strengthening some key players view concerning the need for cloud audit capabilities, namely in the context of the development of standards for the federation of cloud audit data. Clearly cloud audit is an unavoidable issue that is present in the agenda of standardization bodies (e.g. CSA, DMTF, TOG), associations (e.g. EuroCloud) and governments (e.g. USA and the European Community countries).

Interoperability and standards for cloud management are important issues that must be addressed to support cloud federation. To this end, several standardization organizations are currently working, namely the Distributed Management Task Force (DMTF) [15] and the Open Grid Forum (OGF).

In the years 2009 and 2010 DMTF published informational specifications (Cloud Incubator) concerning cloud interoperability, architecture and interfaces, use cases and resource interaction model. Recently, the Cloud Management Working Group (CMWG) replaced Cloud Incubator initiative in pursuing the writing of formal specifications, concerning the cloud management logical model and a Representational State Transfer (REST) protocol mapping. Such work is focused on IaaS and leverages on other standards, like the Open Virtualization Format (OVF), OVF [18] is a XML-based packaging format for virtual appliances, it relies on metadata for artifacts description (using DMTF’s Common Information Model). According to DMTF, some of its benefits are: deliver software through portable virtual machines, streamlined installations, virtualization platform independence and flexibility.

Open Cloud Computing Interface (OCCI) is a OGF initiative that published a RESTful protocol and API “for all kinds of management tasks.” Originally OCCI allowed many of the common remote management tasks for IaaS model, namely deployment, autonomic scaling and monitoring [21,22]. According to OCCI, its current release has “a strong focus on integration, portability, interoperability and innovation while still offering a high degree of extensibility.” OCCI working group expects that its API shall conveniently address PaaS and SaaS layers.

Finally, cloud monitoring and control management must be envisaged to be interoperable with current management technologies and standards. Cloud computing will benefit from interoperability with mainstream and emerging Network and Systems Management (NSM) platforms [5]. Why? First, promoting the integration with mainstream management domains, Cloud computing Frameworks (CCF) will profit
from widely deployed management standards and widespread knowledge regarding their use. Secondly, CCF will capitalize from emerging management technologies and tools, which address contemporary management requirements. Thirdly, NSM will offer a common interface to unify cloud frameworks monitoring, configuration and event handling. Lastly, it provides a feasible path towards the integration of CCF management with host (physical) and virtualize resources management.

4 Cloud Computing in Higher Education

Nowadays European countries are facing economic difficulties that will probably lead to the contraction of HEIs’ public funding. To keep their competitive edge, HEIs must reduce their budget without compromising their Quality of Services (QoS). To address such stringent requirement, a feasible solution is to deploy their administrative, academic and research ICT services into the cloud. Such solution is attractive from the economical point of view, and it appears to be more efficient and productive [27].

Porto Polytechnic shares with other institutions of higher education the desired to embody fully the three sides of the “knowledge triangle”: Higher education, research and business-innovation. Being a founder member of EuroCloud Portugal[2] [17] is an excellent opportunity to layout partnerships with others that share a common interest in conceiving the next generation of educational ICT solutions based on free or low-cost technology, low development and deployment time, quick adaptation to the required demand, flexibility toward content diversity and client heterogeneity, and low resource management effort. Back in 2006, Porto Polytechnic embraced Cloud computing concept through the introduction of SaaS principles in software disclosure. Following this effort, it was the first Portuguese higher education institution to use Cloud computing technology for e-learning and workshops—the Comunidade.EU initiative. Currently, 4,800 Porto Polytechnic members work on a daily basis with this platform for, among others, document sharing, application development, email and calendar services. The Comunidade.EU main goals were to reduce on premises management effort, the investment in physical servers, licenses and technicians, as well as increase the QoS by taking advantage of Google Apps platform. Likewise Porto Polytechnic, many (about 20% according to Educause) North American HEIs are already using SaaS model for email system, and some of those are evaluating the benefits of migrating to Google Apps.

Amazon Web Services (AWS) is a major Cloud computing provider, offering to the worldwide academic community the benefits of AWS for teaching and researching—the AWS Education program. According to Amazon [6], “educators, academic researchers, and students can apply to obtain free usage credits to tap into the on-demand infrastructure of Amazon Web Services to teach advanced courses, tackle research endeavors and explore new projects—tasks that previously would have required expensive up-front and ongoing investments in infrastructure.” Since 2007 that University of California at Berkeley (UCB) students are exposed to Cloud computing technology through the development and deployment of SaaS applications. From 2008, these projects are being deployed on the Amazon Web Services (AWS) cloud. The UCB reports [19] that

Courseware management was made easier by AWS’s pervasive use of virtual machine technology: we created a single virtual machine image containing the whole software stack, and each student or team could deploy that image on an EC2 server instance and instantly have the same experience as if they themselves were administering the application server in a datacenter. We would never grant undergraduates root access on a shared Berkeley server, but with EC2 they can have root access on their own image, and any damage they do can be undone by simply reinstantiating the image on a new server.

Other initiatives include virtual machines on-demand through community clouds offering IaaS services. Such is the case of Science Clouds informal group 3. This group is composed by the Universities of Chicago (UC), Florida (UFL), Purdue, and Masarky, each one operating its own site: three in the US and one in Europe. Every site has a different setup and is maintained with the support of a cloud research project: Nimbus (UC), Stratus (UFL), Wispy (Purdue) and Kupa (Masarky). The cloud test bed is made available through a Nimbus cloud client (as each site is based on Nimbus CCF). Resources aren’t shared among sites as no common policies are applied to resource pooling allocation. As Science Clouds promoters warn, this initiative is “a beta project deployed on a modest allocation of resources.” Nonetheless, an important one, that if further developed can provide a new trend that HEIs can pursue to develop and deploy their ICT infrastructures within a federation of clouds. Such voluntary partnerships between HEIs that share common interests shall build stronger bonds, promoting their collaboration and the mobility of their students and staff.

5 An European Federation of Academic Clouds

Despite limitations of today’s e-learning platforms they are recognized to enhance learning and support non-traditional learning paths [1]. Although such learning platforms and course contents are eventually shared by HEIs with their peers, and with society in general, being MIT open courseware initiative4 a notable case, commonly such resources are privately administered and for private use. This reveals a shy openness between HEIs when it concerns knowledge and resources sharing. Such practice opposes to EHEA successful deployment. Cloud computing is an opportunity to address such problem, as it provides technological solutions to share infrastructures, applications, services and data. Cloud technology enhances the ability to cooperate, speed up processes, increase services availability, and resources scaling with potential reduction costs. The deployment of HEIs private clouds and their steady migration to a hybrid model federated space of academic clouds present as a feasible path to support technology heterogeneity while enforcing cloud interoperability.

A federation of European academic clouds would involve higher education institutions with different backgrounds that, eventually, had no prior contact with Cloud computing technology. Such multidisciplinary cooperation between partners would be a strong point: The input of participating institutions does not originate in one specific area but it rather comprises different areas, bringing together institutions that had not

3 http://www.scienceclouds.org
4 http://ocw.mit.edu
had joint projects and promote cooperation between different areas. It is the author’s belief that this would open a whole new trend of national cooperation and it would be an important step toward the implementation of Bologna mobility and cooperation spirit.

The following analysis is aimed at identifying the added value of Cloud computing concerning HEIs globalization on information sharing, and student and higher education staff mobility. These use cases follow the impediments identified by Campbell [13] and reflect the need to think higher education on a global scale. Campbell’s work (still an ongoing effort) aims the establishment of a new CDIO (Conceive, Design Implement and Operate) standard, the 13th, named “Internationalization and Mobility”. CDIO initiative has adopted 12 standards that address engineering curricula development, design-build experiences and workspaces, modern teaching methods, new assessment procedures and evaluation methods [16].

5.1 Globalization: Information Sharing Challenge

HEI globalization allows students and higher education staff to acquire the demanded proficiency to ensure their success in a global workplace. Such competences are no longer confined to scientific and technical issues, but include language skills, as well as social, cultural, political and ethic knowledge. Cross-credit processes and international dual awards are within initiatives that HEIs are already deploying and which required technological support to make such efforts effective. Cloud computing SAaaS services might present feasible solutions to this use case, as it is required to develop common interfaces to promote interoperability among HEIs’ academic/administrative applications and information systems. One area that must be addressed is the enhancement of the European Authentication and Authorization Infrastructure (AAI) to support secure academic information transactions using standard procedures related with metadata description and information mapping, authentication, and data confidentiality [4]. It is likely that some of these open issues can be challenged using Semantic Web and Policy management standards, in what concerns the enrichment of information description, data consolidation, account management interoperability and Service Level Agreements (SLA).

5.2 Cooperation and Mobility Promotion

Internationalisation requires a steady flow of financial support for institutions and mobility scholarships. Eventually, in years to come, this can reveal to be problematic for South European countries like Portugal. According to OECD (Organisation for Economic Co-operation and Development), overall funding per students in OECD countries “has slowed down since the early 1990s” [25]. The same study concludes that direct public funding in 2003 was still the main source of revenue for most of the European public HEIs, namely Portuguese (about 90%), while in Asia/Pacific such scenario was quite different. In 2003, Japan, Republic of Korea and Australia direct public funding was less than 40%, being the remaining at household’s expenditures. In OECD countries private funding has a small impact on HEIs budget, except in the United States, but it has grown 5% since early 1990s to 2003. Despite these facts, internationalisation
is not only about costs, but an investment that provides important direct revenues: Australia international students (in-bound) accounted, in the academic year 2007-08, for the third place in the export balance ($14.1 billion).

The trend should be the establishment of partnerships, and through them improve HEIs portfolio, attract more foreign students, and reduce operational costs by sharing “academic commodities.” It is in this context that Cloud computing paradigm can make a difference: Datacenters consolidation, cluster resources sharing, and the usage of third party (business) clouds for academic services (email and others) shall allow, in the mid term, financial advantages as costs associated with software and hardware acquisitions (on-site installations), and technical staff are reduced. The University of Westminster reports (Sultan 2010), “The cost of using Google Mail was literally zero. It was estimated that providing the equivalent storage on offer on internal systems would cost the University around £1,000,000.”

To gain economical advantages from Cloud computing, HEIs can, on one hand, start to use free/low cost services provided by business (education programs), on the other hand, migrate its monolithic datacenters to the cloud (private). However, this is just “the tip of the iceberg.” The deployment of a cloud community whose members openly cooperate in a cloud federation, assuming both the provider and consumer roles, that supports transparent and elastic provision allowing the dynamic scale up/down of HEI’s resources (IaaS), lead to the full potential of Cloud computing paradigm. In this case, each federated HEI should take the provider role and contribute to a common resource pool, accepting common management and control policies, deploy common provision rules and agree with SLA principles. A federated identity providers must be established, similar to the AAI platform deployed to support the federated space of Learning Management Systems (LMS).

6 Cloud@RCTS Proposal

Back in 2006 Porto Polytechnic embraced Cloud Computing concept through the introduction of SaaS principles in software disclosure. The main goals were to reduce on premises management complexity and increase QoS. At that time, two platforms—Microsoft Live and Google Apps—were elected for requirements analysis, conformance and performance assessment. Such evaluation focused on management, provisioning, interoperability and integration with Porto Polytechnic platform for authentication, authorization and access control: Lightweight Directory Access Protocol (LDAP) based platform, named DirIPP. Such platform was designed according to RFC2247 and integrates staff and students’ access profiles, including password management, for a wide-range of applications and services. Despite developers’ efforts to offer a Single-Sign-On (SSO) mechanism, the system had some interoperability shortcomings with third party platforms, namely Google Apps. To provide full integration of Google’s platform and Porto Polytechnic DirIPP, the SSO mechanism was redesigned according to the Security Assertion Markup Language (SAML) 2.0 standard. This XML-based standard is supported by Google Apps (as the service provider) and secures the exchange of user authentication and authorization data with the identity provider, the Porto Polytechnic. Deploying SAML-based SSO was not an easy task, due to the state of the art
of open source SAML 2.0 projects, namely Shibboleth\(^5\) and SimpleSAML\(^6\). Even so, Porto Polytechnic developers, lead by Paulo Calçada, decided to implement a Web SSO service compliant with SAML 2.0.

The Cloud@RCTS proposal (submitted to FCCN in July 30th 2010 [12]) is the next step in the effort to deploy Cloud Computing services in the IPP and other Portuguese HEIs, including R&D centers and the Portuguese GRID computing platform. From the author’s perspective, the very essence of Cloud Computing for educational purposes are to innovate on ICT-based content, services, pedagogies and practice for lifelong learning. We will increase collaborative work, information availability, reduce the stages for transmitting information and thus increasing reliability of the information provided, reduce the cost of access to information and increase the proximity between teachers and students. It is e-learning and e-teaching taken to the next step and next decade.

Cloud@RCTS intends to the development a heterogeneous structure for sharing resources and processes. Such aims imply finding the right solutions to some fundamental issues: resource optimization and costs improvement of maintaining and operating infrastructures, increase service quality, standardization of processes and technologies, processes streamlining and simplification.

![Fig. 1. Portuguese NREN (National Research and Education Network)](image)

\(^5\)http://shibboleth.internet2.edu
\(^6\)http://rnd.feide.no/simplesamlphp
Cloud@RTCS proposal is focused on the SaaS and IaaS layers and it is based on the nationwide gigabit data network (Figure 1\(^7\)). Such infrastructure connects Portuguese research and higher education institutions and between these and the pan-European R&D communities through Géant network (www.geant.net). Concerning PaaS, the proposal authors wish to join synergies with existing infrastructures within the GRID community. Following the e-Science European initiative, the Portuguese institutions currently providing Grid Computing services shall be integrated in this virtualized infra-structure. The cohabitation of native Grid applications and resources in Cloud Computing Ecosystem anticipates the integration of grid resource management in the Cloud@RTCS platform.

6.1 Reference Model for IaaS Layer

In what concerns IaaS level, the authors’ proposal is to deploy a set of virtual platforms providers, based on each organization contribution to the global cloud, emerging a set of heterogeneous clouds resulting from the migration of monolithic datacenters to a new concept: An integrated, elastic, fault resilient and secure cloud service providers. Business providers are envisaged in our proposal; since common APIs are agree on. Inevitably this scenario presents many challenges, being some of them interoperability and context compatibility, as discussed in [23]. Its is envisage that RCTS infrastructure own a set of (common) resources, geographically distributed, to which all parties have access according to specific usage policies.

Such hybrid model (Figure 2) results from the federation of private, community or public clouds would improve the physical/logical resource sharing efficiency among institutions. The scheme suggests the existence of a high-level intermediate entity (meta-broker) responsible for the coordination between institutions own brokers. Such scenario describes HEIs’ resources sharing in the context of a federation of clouds. It will be the meta-broker responsibility to trade with external suppliers, eventually others federations. Arising from the geographical distribution of resources and their mobility, the meta-broker, in coordination with the local brokers, should implement roaming mechanisms.

This part of the proposal identifies three main activities that involve the analysis, specification and development of prototypes for the meta and local brokers and the interconnection with third party providers (trade). The first two activities relate to the areas of scheduling, provisioning, monitoring and control. The last one will be on the development of adapters that allow trading with external resources (from the federation point of view) transparently to the user.

6.2 Reference Model for SaaS Layer

In what concerns SaaS layer, the proposal includes the interoperability of application and services through a set services including (meta) data mapping, authentication and data confidentiality (Armbrust 2009). To ensure interoperability with existing services

\(^7\) Credit notice: NREN map, dated April 2009, available at FCCN site (http://www.fccn.pt).
(e.g. public services) it is proposed the use of the Portuguese Agency for Administrative Modernization (AMA) common services framework. The suitability of such framework to the RCTS universe is still to be verified. Figure 3 displays a feasible scenario where a message exchange between User A and C relies on a third user (B) to implement transaction control. The depict message exchange stresses the importance placed on identification and digital signature mechanisms inherent in all actions related to the flow of information. However, the existence of three entities serves the case study described in Figure 3. The platform architecture should be independent of the processes and entities involved.

It is likely that some of these open issues can be challenged using Semantic Web and Policy management advances, namely in what concerns the enrichment of resource description, management interoperability and Service Level Agreements (SLAs). It is the author view that a framework to automate the translation between data formats will be needed. Such framework could allow the storage of native (meta) data in RDF/OWL triple stores, promoting data consolidation and knowledge inference.

7 Conclusions

This keynote proposes the development and deployment of a federated space of academic clouds as it appears to be a feasible solution to promote the establishment of the
European Higher Education Area according to the Bologna Process spirit: cooperation, internationalization and mobility. Such proposal addresses cloud heterogeneity and endorses interoperability. Business providers can be envisaged in this proposal, since common interfaces are agree on. Inevitably this scenario presents many challenges that, at the time of this writing, there aren’t adequate solutions. Fortunately, active research programs are addressing some of the key challenges. Buyya [9] presents a study that “demonstrates that federated Cloud computing model has immense potential as it offers significant performance gains as regards to response time and cost saving under dynamic workload scenarios.” OpenNebula (www.opennebula.org) is working on a prototype to address cloud burst, allowing the cloud primary site to scale up using secondary sites resources. Also, Keahey (2009) proposes Sky computing to support “dynamically provisioned distributed domains” built over several clouds. While, Bertino [8] addresses the need of develop flexible access control mechanisms that support privacy-preserving digital identity management for Cloud computing infrastructures. Finally, standardization organizations like DMTF are currently addressing key issues like cloud management, interoperability and security.

The Cloud@RTS business principles are still an open issue, however the peer-to-peer model presents as a feasible one: The federation resources that each institution can use depends on the resource volume that they are willing to share. It is expected that many of the federated resources come from the institutions existing computing resources. In this case, it will be necessary to establish “admissibility requirements” as to level the expectations of all parties concerning the federated resources performances.
Besides the number of CPUs, memory and storage capacity, and networking, also issues like datacenter certification can play a major role.

References

25. K. Kärkkäinen. Emergence of Private Higher Education Funding Within the OECD Area, 2006. OECD.
Opening A Whole New World with HTML5

Tiago Andrade e Silva
Microsoft Portugal

Abstract. HTML5 is the biggest and most powerful HTML specification to ever come out of the W3C. Its roots come from a proposal to extend HTML4 to make it easier for web developers to create web applications. With more and more browsers supporting HTML5, ECMAScript 5, SVG and other web standards, developers now have a strong web platform they can use to create a new class of web application that is more powerful and interactive than ever before. What’s in HTML5 that lets us take our sites to the next level?
Full Papers
An XMPP messaging infrastructure for a mobile held security identity wallet of personal and private dynamic identity attributes

Alexandre B. Augusto, Manuel E. Correia
aaugusto@dcc.fc.up.pt, mcc@dcc.fc.up.pt

Center for Research in Advanced Computing Systems (CRACS-INESC LA)
Department of Computer Science, Faculty of Science, University of Porto

Abstract. We are currently witnessing the widespread usage of personal mobile smart devices with serious practical computational power and Internet connectivity. The most popular of these devices being the smartphone, which is in its way of becoming an ubiquitous powerful personal device. At the same time sites like Google and Facebook are deploying an ever increasing set of personal services that are being aggregated and structured over personal user accounts were an ever increasing set of personal private sensitive attributes is being demanded as an excuse for better services interoperability. These user attributes are extremely valuable for these global Internet service companies, as they allow them to produce highly accurate user profiles that they can then monetise very efficiently for marketing purposes. The more accurate a user profile is, the more valuable it becomes, and there are certain kind of personal attributes these companies have just started to harvest that present a major threat to personal security and privacy. These attributes are highly dynamic and are intimately associated to their owners by the means of their personal devices. One example is the user GPS position and other dynamic attributes like heart beat, body temperature, etc... that can be collected and maintained by the user personal mobile devices, sometimes without the user being aware of their disclosure into highly sensitive personal profiles.

In this paper we propose and describe an identity management framework that allows users to asynchronously control and effectively share this type of sensitive dynamic data thus guaranteeing security and privacy in a simple and transparent way. Our approach is realised by a fully secure mobile identity digital wallet, running on mobile devices (Android smart phones), where users can exercise discretionary control over the access to sensitive dynamic attributes, disclosing their value only to pre-authenticated and authorised users for determined periods of time. For that we rely on an adaptation of the OAuth protocol to authorise and secure the disclosure of personal private user data and new XML Schemas to establish secure authorisation and disclosure of a set of supported dynamic data types that are being maintained by the personal mobile digital wallet. The communication infrastructure is fully implemented over the XMPP instant messaging protocol and is completely compatible with the public XMPP large messaging infrastructures.
1 Introduction

The massive aggregation of personal identity attributes is currently one of the most important structural and strategic endeavors currently being carried out all over the Internet. Global Internet companies like Google and Facebook are ever more competing over personal user data due to its high strategic commercial value on the market [5], making user digital identity a strategic asset that is going to help to redefine what kind of new innovative services are going to be developed and how they are going to be deployed all over the cloud in an interoperable way. This is well illustrated by the current fierce competition being fought by these companies about digital identity and their associated authentication and data exchange protocols like OpenID [15] and OAuth [7].

Currently, identity attributes are normally composed by static values held in the identity management system running in the cloud which can be a bad idea [13]. What we intend to do with the work described in this paper is to expand the universe of managed static identity attributes with dynamic values that by their very nature are more intimately associated with their owner and therefore can only reside, not in the cloud, but in mobile smart devices that follow their owner everywhere and can therefore keep those values up to date. One good example being the GPS coordinates.

What we are proposing is to expand the set of current static attributes being managed and held by Internet identity management systems (IIMS) [18] with a new set of highly dynamic changing attributes. These new identity attributes can be instantiated in IIMS as symbolics names that can act as pointers to their real location in the Internet and that allow for the Relying Party (RP) to locate the digital attribute storage wallets (DASW) where those dynamic attributes are being maintained and protected. In this highly dynamic identity infrastructure we are currently developing (Open Federated Environment for the Leveraging of Identity and Authorisation - OFELIA), every time a RP wants to consult the current value of a certain dynamic attribute it has first to locate the DASW where it resides and then ask its owner for permission to access its updated value for a certain period of time. The attribute owner then has the power to allow or deny that request and provide the RP with an OAuth authorisation token, that the RP will then use and present as proof of previous authorisation every time he wants to monitor the dynamic attribute during the previously authorised period of time. The attribute owner maintains revocation rights by being able to remove access at any given moment.

It is important to realise that dynamic identity attributes constitute a whole new concept of digital identity because their value is constantly being changed and to be of any value, the RP has to constantly be able to monitor it as needed. This is easily illustrated by the GPS location. With this attribute the RP can obtain the real near current time position of an individual and not the last time the user or application remembered to update it.

Our major motivation for OFELIA is to create a communication infrastructure based on public XMPP infrastructures, network services, applications and API libraries to allow for sensitive information [17] like a GPS position or a med-
tical information like a person heartbeat to be exchanged in a secure, reliable and owner controlled way. Keeping this kind of real time sensitive attributes secure and private between the requester and the data owner is a challenge, specially if the framework makes heavy use of a public XMPP messaging infrastructures like the one currently being operated by Google. We also use Android smart mobile phones as our identity digital wallets. Java is native to the Android operating systems and this allows us to have rapidly built a running prototype for user controlled GPS positioning by taking advantage of the numerous libraries, development systems and applications servers which already run on Android. We have used a Java OAuth library [14] to handle the authorisation process to data, a XMPP messaging server [2] to exchange messages between the requester and the data owner and new OFELIA XML Schemas to validate the authentication, authorisation and identity messages exchanged between RP and the Digital Identity wallet on the mobile device [4].

The rest of the paper is organised as follows. In Section 2, we review the system architecture, describing each node, their functionality and how data flows between the different actors involved. In Section 3 we describe a case scenario that can be quite useful and helps to illustrate the messages exchange that has to occur between the nodes to accomplish the task at hand. In Section 4 we described what was accomplished, some preliminary conclusions for the work we have developed thus far for OFELIA and our plans to the next steps.

2 Architecture

In this section we describe the main components of the OFELIA architecture and discuss the reasons behind some of the options and compromises we had to take to make our vision work in the mobile world. We also take time to describe the flow of data and their important aspects like the protocols and services we have used to build our current secure communication infrastructure. For this stage of development we limit the OFELIA architecture to the development of two types of services, one for the attributes requester (RP) and other for the data owner (Digital wallet or Endpoint), assuming that the data and its owner are both engaged in the same mobile phone. Figure 1 shows our architecture.

![Architecture Diagram](image-url)

Fig. 1: Architecture
2.1 XMPP: Extensible Messaging and Presence Protocol

XMPP is an open technology for real-time communication that uses the eXtensible Markup Language (XML) as a base format for exchanging information encapsulated into small pieces of XML. These XML pieces are sent from one entity to other [16], using an appropriate application level transport protocol, normally HTTP, through the means of a rendez-vous XMPP server that relays these messages to the end-points engaged in communication. XMPP servers provide a standard set of services that can be used by client applications. In OFELIA, our messaging infrastructure relies on the four following XMPP instant messaging core services [d]k the ruthentication giving high confidence that entities attempting to communicate over the messaging infrastructure have been authenticated by the XMPP server in a secure way, the Presence grants OFELIA the capacity of checking entity communication availability allowing the infrastructure to make different decisions based on entity availability, the One-to-one messaging allowing the exchange of peer-to-peer XML messages between RPs and Endpoints and the Contact Lists allowing OFELIA participants to manage a list of trusted entities and thus help a peer to authorise and verify the other peers availability and trustability.

Arguably, in the mobile world, there is some difficulty in directly addressing and communicating with an Internet enabled mobile device. In the mobile world an implicit direct communication with the device is almost impossible due to the shortage of public IPs addresses faced by Internet service providers. In the near future IPv6 is supposed to solve this problem, however it is our strong belief that the mobile Telecommunications operators (Mobicomms) will still not allow for this kind of direct communication to mobile phones due to their very inflexible business plans, where the mobile phone is nowadays mostly regarded simply as a consumer device, not a provider of services. A neutral rendezvous point on the Internet where both RP and Digital wallet can both meet is thus obviously necessary. Towards this end, XMPP is proving to be an almost ideal communication infrastructure for OFELIA because of its core services. Namely i) almost real time messaging, essential to maintain our dynamic data types ii) its ability to operate over HTTP connection by the means of the BOSH(Bidirectional-streams Over Synchronous HTTP) [9] protocol, which allow us to bypass the connectivity problems imposed by the overly restricted mobile Internet access. iii) its capacity to store and forward messages in case any of the nodes becomes offline, which is proving to be essential for asynchronous communications. iv) its scalability to avoid bottleneck problems and the fact that it is a mature fully supported and approved Internet standard, widely deployed and an important part of the communication operations and infrastructure of large Internet operators like for example Google and Facebook.

2.2 OAuth: Open Authorisation

OAuth is a protocol that provides a standard method for clients to access server resources on behalf of a resource owner. It also provides a process for end-users
to authorise third-party access to their server resources without sharing their credentials, using user-agent redirections [7].

There are three actors involved in an OAuth transaction: The data owner (User), a third party web application (TP) and the user data storage (UDS). Usually a User wants to provide a TP with an authorisation to access his data that resides on a certain UDS. To achieve this, the TP redirects the user to the UDS with a formalised request where the user is asked to authorise it. After authorisation the UDS returns to the TP signed authorisation tokens that allows the TP to access the requested data. The security processes involved in the creation and management of authorisation tokens relies on valid digital signatures and on a shared secret between the OAuth consumer (TP) and the OAuth provider (UDS), a process fully explained and described on [6]. Currently, in OFELIA both the User and the UDS join and communicate locally on the same mobile device. Thus, in this case, OAuth communication security is built upon the TP digital X.509 certificate and on a session unique key established between the TP and the UDS (Digital Wallet).

This co-location of both the User and the UDS on the same device also have some deep implications in the way the authorisation request and granting process is managed by the means of the OAuth protocol. Since in OFELIA, both User and UDS meet and are located in the same node (the mobile device), when the TP requests access to some identity attribute, an authorisation request appears on the user node showing on whose behalf the authenticated TP is making an access request, what attributes are being requested and for how much time that access must be provided. The User then has to decide whether to grant authorisation, and this can be done in an asynchronous way. Once the authorisation is granted, OAuth will generate and share an access token and a token secret with the TP that must then be presented every time it wants to consult the authorised user identity attributes. This continues until the OAuth tokens expires or is revoked by the User on the UDS.

2.3 OpenID

OpenID is an Identity Manager [15] that allows a user to sign into multiple websites, with the same account and at the same time control how much of that account user OpenID identity attributes can be shared with the websites he visits. Every time a user tries to authenticate into a website (Relying Party (RP)) with OpenID he is redirected to his OpenID provider where he is then asked to login and authorise the identity attribute exchange requested by the website (RP), after which the user is again redirected to the originating RP.

In order to standardise and define appropriate semantics for a useful set of user attributes that could be universally recognised by all RPs, the full set of standardised and widely recognised identity attributes for OpenID is substantially reduced. This decreases the usefulness of the protocol and have so far limited its deployment almost exclusively to the authentication domain.

In OFELIA we employ OpenID as an authenticator for the RP (Requester) service. Both the user account at the Requester (RP) and the User at the UDS
have registered OpenID identities that are used in OFELIA to authenticate both identities on the RP. This provides a common account creation and registration process that allows both endpoints to have a common and coherent way of acquiring and verifying identity data.

2.4 Requester web service (RWS)

The requester web service is an integral component of the RP and is composed by a database, an OpenID consumer and a XMPP HTTP connector. It uses two XML Schemas to authenticate and validate XMPP communications with the UDS and at the same time maintain appropriate data semantics. It also employs two X509 certificates, both emitted by a common trusted PKI, as a way to assure both endpoints (RP and UDS) identity and establish session keys.

When on behalf of a requester, an OFELIA RP tries to access someone's identity attributes, held on an OFELIA digital wallet, the requester is first asked to login and authenticate himself using an OpenID account. If this is the requester first login on the RP, this action initiates an auto-enrolment process where the RWS stores, in its database, the requester OpenID address, name, jabber address, user certificate and mobile number, if they exist as OpenID attributes. This account information can then later be enriched with OFELIA OAuth tokens for some identity attributes being held in remote digital wallets, owned by this same requester or somebody else. If the requester has already been enrolled into the RP, he is just authenticated via OpenID and his OpenID identity attributes can then be transparently updated. After login, the RP, on behalf of the current user, can request and try to get identity data from a remote Digital Wallet by the means of the digital wallet endpoint jabber address. If this jabber address is not yet registered onto the RWS, a XMPP message is sent to this address, requesting registration. If the digital wallet jabber endpoint is not reachable, the RWS can nevertheless send an asynchronous request authentication to the digital identity wallet jabber address. This message is held by the XMPP communication infrastructure until the digital wallet comes on-line. If there is a mobile phone number available for the digital wallet, the RP can send a SMS message to the smart phone where the digital wallet resides, requesting this endpoint to have his digital wallet to connect into the XMPP infrastructure in order to receive the pending OFELIA requests that have been sent by the RP on behalf of the requester.

The security details about authentication between the endpoints and the data exchanged are explained at subsection 2.7 and exemplified at section 3 on subsection 3.1 and 3.2.

2.5 Endpoint Web Service (EWS)

The Endpoint service was developed to be deployable on mobile devices and must take into consideration that the data owner has to personally intervene as a human, during the authorisation process. The EWS is currently composed by a database, both an OAuth consumer and provider, a XMPP connector, one X509
The digital wallet at the vWS must be logged into the XMPP infrastructure with its jabber id and then wait for OFELIA access requests. When a request is eventually received, the EWS must validate it against an appropriate XML Schema and process it. In case the requester does not exhibit a valid OAuth token, the digital wallet owner will be asked, by the means of an appropriate GUI, to authorise or deny the access request. All granted tokens are stored in a database at the EWS along with identity information of whom they have been emitted to, together with an expiration date determined by the wallet owner. Thus, in order for the EWS to receive OFELIA requests from an authenticated RWS, it is mandatory for the RWS to have had a remote user authenticated by OpenID, on behalf of which the OFELIA request is being made and provide the EWS with the relevant identity information needed by the wallet owner to make an informed authorisation decision.

The security details about authentication between the endpoints and the data exchanged are explained at subsection 2.7 and exemplified at section 3 on subsection 3.1 and 3.2.

2.6 XML Schema

We use OFELIA XML Schemas to help maintain system interoperability between services and be able in the future to decouple endpoint services for different OFELIA implementations[12]. We employ two different OFELIA XML Schemas, one to handle authentication processes (OfeliaAuth) and the other (OfeliaDataEx) to handle the data exchange that takes place when a RP reads attributes from a digital wallet.

In section 3 we present in more detail an XML exchange flow to better elucidate the documents interchange that can occur in the OFELIA identity infrastructure.

**OfeliaAuth** As can be seen in figure 2, the XML Schema used for authentication, consists of a root element *OfeliaAuth* composed by three sub-elements: *Header*, *User* and *KeyAuthentication*.

The *Header* element carries information about the state of the authentication and the type of the OFELIA request. The *User* element contains personal user information: a Jabber id and an OpenId identity to allow EWS verify user requester identity and RWS Public key to exchange a session symmetric key in a secure channel. The *KeyAuthentication* element is composed by a challenge string, ciphered with the Endpoint public key, and a blank attribute used to return a ciphered session symmetric key.

**OfeliaDataEx** As we can see on figure 3 the XML Schema employed for identity data exchange consists of a root element *OfeliaDataEx*, once again composed by three elements: *Header*, *User* and *Data*.
At the Header element we keep information about the state of the data exchange and the type of the OFELIA request. At the User element we have the user Jabber id, OpenId and another element, Tokens, composed by three attributes: AuthorizationToken, TokenSecret and ExpireDate. The Data element is composed by optional elements, describing the nature of the dynamic identity attribute being described. Currently we have a gps element defined with a latitude, longitude and a timestamp. We are currently defining several other elements to describe other dynamic attributes like heart beat, blood pressure, etc., that could prove to be useful for remote monitoring web applications. The Data element can thus contain highly diverse types of formalised dynamic data types, to cover a Highly diverse range of application areas. In other words, we can provide for all kind of personal dynamic attributes so long as its data type is formalised in the OfeliaDataEx XML Schema. It is also mandatory that all dynamic type elements have a valid timestamp attribute, not only to be able to maintain an historic value for its values but also to prevent the resending of the same value during different data exchanges.
2.7 Data flow

OFELIA interactions are divided into two main different operational phases. The first handles the authentication process, and it works by exchanging a session key, using a challenge-response cryptographic scheme, that is then employed to set a secure tunnel between the RWS and the EWS. The second phase handles the creation and management of OAuth access tokens and the subsequent identity data consultation.

Assuming both web services have already been logged on into a trusted XMPP messaging infrastructure and that at every step, data is validate by the appropriate XML Schema, the data flow for authentication accomplished by the following three steps:

1. The RWS generates an OfeliaAuth XML, sets a challenge string ciphered by the EWS public Key from the mandatory previous register on RWS and completes the others attributes leaving only the Key empty. It then sends the filled document to EWS by XMPP as can be see on figure 4a under subsection 3.1.
2. The EWS verifies the requester data, deciphers the challenge and ciphers it again with the RWS public key received on XML. It then generates a session key to set on the attribute Key and sends it back to the RWS, cyphered with the RWS public key. This is illustrated on figure 4b under subsection 3.1.
3. The RWS deciphers the challenge and if is valid it stores and sets the received attribute Key as a session key to secure the subsequent communication phase of OFELIA.

The data flow for OAuth tokens management and data exchange can accomplished in six steps. Four steps for OAuth token management and two steps for identity data consultation. These six steps are ciphered and deciphered with the established session key obtained in the previous phase, resulting a secure channel of communication between the RWS and EWS.

1. The RWS presents an OfeliaDataEx document, with a valid User element and with an appropriate Header to request the identity data and sends it by XMPP to the EWS. This is illustrated by the figure 5a under subsection 3.2.
2. On receiving an OfeliaDataEx document, the EWS updates the header, sets the data types available with an empty value and sends it back to the RWS. This is illustrated by the figure 5b under subsection 3.2.
3. On receiving the partially filled OfeliaDataEx document, the RWS updates the header, deletes any identity attribute it does not want to consult, sets the attribute ExpireDate and sends the updated OfeliaDataEx document to the EWS. This is illustrated by the figure 6a under subsection 3.2.
4. On receiving an OfeliaDataEx document with the the attribute ExpireDate set, the EWS uses the information provided by the document to ask the owner of the identity attributes whether he authorises the consultation of these attributes by the entity whose identity is described in the OfeliaDataEx document. If the authorisation is to be granted the EWS then updates the
header, generates OAuth tokens for the requested attributes and puts then into the AuthorizationToken and TokenSecret attributes. It then sends the updated OfeliaDataEx document back to RWS. This is illustrated by the figure 6b under subsection 3.2.

5. On receiving an OfeliaDataEx document with OAuth tokens, the RWS updates the header, stores the tokens, set the timestamps with the last timestamp received for the requested data and sends the just updated OfeliaDataEx document with OAuth tokens back to the EWS. This is illustrated by the figure 7a under subsection 3.2.

6. On receiving an OfeliaDataEx document with OAuth tokens and a timestamp, the EWS updates the header, verifies the validity of the presented OAuth tokens, and fills the data element of the OfeliaDataEx with an array of dynamic type elements with the historic set of values the dynamic type has assumed on the Digital Wallet since the timestamp just received on the OfeliaDataEx document. It then sends the document back to RWS by XMPP. This is illustrated by the figure 7b under subsection 3.2.

3 Usage case Scenario

In this section we are going to describe a concrete case scenario to help better clarify and better illustrate how the OFELIA XML Schemas can be used within the XMPP infrastructure for a RP to get access authorisation and then subsequently present authorisation tokens to monitor dynamic attributes from a remote digital wallet held on a mobile device. For illustration purposes we are going to explore a real case scenario where a web application is authorised to monitor a user by the means of the GPS device he has on its mobile phone. Let us assume that a taxi company, let us name it “We know where you are”, decides to use our system to implement an innovative service on the Internet for its costumers.

The taxi company releases a mobile application with our endpoint OFELIA web service for smart phones with a GPS. Every time a passenger needs a taxi he uses the application to request the service. The company cloud application on the Internet then asks authorisation, using the OFELIA infrastructure to have for the next quarter of an hour, temporary access to the costumer digital wallet to track the costumers geographic location thus allowing the nearest taxi driver to find him. This solves problems like unknown roads and moving costumers. Since the tracking authorisation is temporary the costumer privacy is protected because that information is only disclosed when the costumer is in need of service and everything is kept under his own control.

For a better comprehension of the data flow we are going to present a simplified XML OFELIA flow of documents for this scenario, the taxi company Internet cloud application acts as an OFELIA RP and the mobile application on the costumer phone acts as an OFELIA digital wallet.
3.1 XML data flow for Authentication:

A customer decides to request a taxi using the mobile application. The taxi company then acts as a requester and sends a document request with a challenge. The mobile processes the XML request and replies with a session key, as illustrated on figure 4.

![XML data flow for Authentication](image)

(a) Request  (b) Response

Fig. 4: OFELIA XML data from authentication method

3.2 XML data flow of data exchange:

After the establishment of an OFELIA session, the taxi company application sends a request to determine what types of dynamic identity attributes the customer holds in his identity wallet. The mobile phone processes the request and sends the XML document back with a response, where the Data entity is properly filled with the relevant information. This process is illustrated in Figure 5.

![XML data flow of data exchange](image)

(a) Request  (b) Response

Fig. 5: OFELIA XML data from Data list request

The taxi company requester now deletes from the XML response every data type supported by the remote digital wallet, except for the GPS dynamic identity attribute. It then sends via XMPP the updated document back to the mobile phone, requesting access OAuth tokens with an appropriate small expiration
date. On reception the customer is asked by the mobile application to accept the request and grant access to the GPS. The mobile then sends the response back to the company application with valid OAuth tokens. Figure 6 illustrates this process.

![XML data from tokens request](image)

**Fig. 6: OFELIA XML data from tokens request**

With the tokens stored on the company application, the taxi company now sends an OFELIA request each 5 seconds to monitor the customer GPS location data. If the OAuth authorisation tokens are correct and correspond previously established session, the customer mobile replies with an XML response with the GPS entity data correctly filled with updated data. This information is then relayed to the nearest taxi driver that can thus temporarily track the customer and find him in the shortest possible time. This can be observed on figure 7.

![XML data from data exchange](image)

**Fig. 7: OFELIA XML data from Data exchange**
4 Conclusions

In this paper, we have described how the OFELIA XMPP communication infrastructure implements an user empowering way of disclosing sensitive dynamic attributes held on mobile devices. The whole process relies as much as possible on standard well established services and protocols, thus allowing for a simpler implementation and rapid service deployment. We are currently developing a working prototype composed of and RP and digital wallet held on an Android phone that uses the Google XMPP messaging service to exchange documents in a secure and private way. The prototype RP tracks the user GPS and plots its location using the google map service for the periods of time requested by the user.

This digital identity infrastructure comes at a time where there is a real need for the users to gain back some control of their privacy and only disclose their most sensitive identity attributes when they need a service from the Internet that really requires access to this data values to work, and this for only a very limited period of time, kept under strict control by the user.

With our OFELIA prototype we have also proved that XMPP, with its very rich set of working extension services, constitutes an excellent choice for communication, allowing us to quickly set up a system ready to manage and implement complex data exchange processes in almost real time. With XMPP problems like connection restrictions, offline messages or security are readily solved in a standard way, as has been previously described in this paper.

4.1 Future work

In the near future OFELIA will have:

- EWS on mobile phone - OFELIA is still running on desktop computers. The next step will be to deploy the EWS directly into a real android mobile phone with a GPS. We have already did feasibility studies and experiments about running web services on Android devices and found the web container called i-jetty [1] to be good choice to run the EWS on a mobile phone.

- More Dynamic Identity attributes XML schemas - We want to expand OFELIA to other application areas. The eHealth sector [11] is one area where there is a real need to remote monitor patients in a secure privacy oriented way. This is currently done with highly expensive equipment with no provision at all for the patients privacy. With OFELIA we want to be able to the same thing in a better way, by taking care of the patients privacy and the same time provide remote monitoring services with much less expensive equipment, mainly composed of Android phones and bluetooth compatible body sensors.

- RWS as an API - To allow for good extensibility, usability and portability of the system it is essential in the future to program the RWS with the help of an easy to use decoupled API for an easy integration of monitoring services into different web applications without the programmer having to
know about the intricate details of OFELIA XML document interchange and OAuth tokens.

Acknowledgments

This work is being financed by the FCT funded project OFELIA (PTDC/EIA-EIA/104328/2008), and is being conducted with the institutional support provided by DCC/FCUP and the facilities and research environment gracefully provided by the CRACS (Center for Research in Advanced Computing Systems) research unit, an INESC LA associate of the Faculty of Science, University of Porto.

References

PExIL: Programming Exercises Interoperability Language

Ricardo Queirós¹ and José Paulo Leal²

¹ CRACS & DI-ESEIG/IPP, Porto, Portugal
ricardo.queiros@eu.ipp.pt
² CRACS & DCC-FCUP, University of Porto, Portugal
zp@dcc.fc.up.pt

Abstract. Several standards appeared in recent years to formalize the metadata of learning objects, but they are still insufficient to fully describe a specialized domain. In particular, the programming exercise domain requires interdependent resources (e.g. test cases, solution programs, exercise description) usually processed by different services in the programming exercise life-cycle. Moreover, the manual creation of these resources is time-consuming and error-prone leading to what is an obstacle to the fast development of programming exercises of good quality.

This paper focuses on the definition of an XML dialect called PExIL (Programming Exercises Interoperability Language). The aim of PExIL is to consolidate all the data required in the programming exercise life-cycle, from when it is created to when it is graded, covering also the resolution, the evaluation and the feedback. We introduce the XML Schema used to formalize the relevant data of the programming exercise life-cycle. The validation of this approach is made through the evaluation of the usefulness and expressiveness of the PExIL definition. In the former we present the tools that consume the PExIL definition to automatically generate the specialized resources. In the latter we use the PExIL definition to capture all the constraints of a set of programming exercises stored in a learning objects repository.

Keywords: eLearning, Learning Objects, Content Packaging, Interoperability.

1 Introduction

The concept of Learning Object (LO) is fundamental for producing, sharing and reusing content in eLearning [1]. In essence a LO is a container with educational material and metadata describing it. Since most LOs just present content to students they contain documents in presentation formats such as HTML and PDF, and metadata describing these documents using Learning Objects Metadata (LOM), Sharable Content Object Reference Model (SCORM) [2] or other generic metadata
format. When a LO includes exercises to be automatically evaluated by an eLearning system, it must contain a document with a formal description for each exercise. The Question and Tests Interoperability (QTI) [3] is an example of a standard for this kind of definitions that is supported by several eLearning systems. However, QTI was designed for questions with predefined answers and cannot be used for complex evaluation domains such as the programming exercise evaluation [4]. A programming exercise requires a collection of files (e.g. test cases, solution programs, exercise descriptions, feedback) and special data (e.g. compilation and execution lines). These resources are interdependent and processed in different moments in the life-cycle of the programming exercise.

The life cycle comprises several phases: in the creation phase the content author should have the means to automatically create some of the resources (assets) related with the programming exercise such as the exercise description and test cases and the possibility to package and distribute them in a standard format across all the compatible systems (e.g. learning management systems, learning objects repositories); in the selection phase the teacher must be able to search for a programming exercise based on its metadata from a repository of learning objects and store a reference to it in a learning management system; in the presentation phase the student must be able to choose the exercise description in its native language and a proper format (e.g. HTML, PDF); in the resolution phase the learner should have the possibility to use test cases to test his attempt to solve the exercise and the possibility to automatically generate new ones; in the evaluation phase the evaluation engine should receive specialized metadata to properly evaluate the learner’s attempt and return enlightening feedback. All these phases require a set of inter-dependent resources and specialized metadata whose manual creation would be time-consuming and error-prone.

This paper focuses on the definition of an XML dialect called PExIL (Programming Exercises Interoperability Language). The aim of PExIL is to consolidate all the data required in the programming exercise life-cycle, from when it is created to when it is graded, covering also the resolution, the evaluation and the feedback. We introduce the XML Schema used to formalize the relevant data of the programming exercise life-cycle. The validation of this approach is made through the evaluation of the usefulness and expressiveness of the PExIL definition. In the former, we use a PExIL definition to generate several resources related to the programming exercise life-cycle (e.g. exercise descriptions, test cases, feedback files). In the latter, we check if the PExIL definition covers all the constraints of a set of programming exercises in a repository.

The remainder of this paper is organized as follows. Section 2 traces the evolution of standards for LO metadata and packaging. In the following section we present the PExIL schema with emphasis on the definitions for the description, test cases and feedback of the programming exercise. Then, we evaluate the definition of PExIL and conclude with a summary of the main contributions of this work and a perspective of future research.
2 Learning object standards

Current LO standards are quite generic and not adequate to specific domains, such as the definition of programming exercises. The most widely used standard for LO is the IMS Content Packaging (IMS CP) [5]. This content packaging format uses an XML manifest file wrapped with other resources inside a zip file. The manifest includes the IEEE Learning Object Metadata (LOM) standard [6] to describe the learning resources included in the package. However, LOM was not specifically designed to accommodate the requirements of automatic evaluation of programming exercises. For instance, there is no way to assert the role of specific resources, such as test cases or solutions. Fortunately, IMS CP was designed to be straightforward to extend, meeting the needs of a target user community through the creation of application profiles. A well known eLearning application profile is SCORM that extends IMS CP with more sophisticated sequencing and Contents-to-LMS communication.

Following this extension philosophy, the IMS Global Learning Consortium (GLC) upgraded the Question & Test Interoperability (QTI) specification [3]. QTI describes a data model for questions and test data and, from version 2, extends the LOM with its own metadata vocabulary. QTI was designed for questions with a set of pre-defined answers, such as multiple choice, multiple response, fill-in-the-blanks and short text questions. It supports also long text answers but the specification of their evaluation is outside the scope of the QTI. Although long text answers could be used to write the program's source code, there is no way to specify how it should be compiled and executed, which test data should be used and how it should be graded. For these reasons we consider that QTI is not adequate for automatic evaluation of programming exercises, although it may be supported for sake of compatibility with some LMS. Recently, IMS GLC proposed the IMS Common Cartridge (CC) [7] that bundles the previous specifications and its main goal is to organize and distribute digital learning content.

3 PExIL

In this section we present PExIL, an XML dialect that aims to consolidate all the data required in the programming exercise life-cycle. This definition is formalized through the creation of a XML Schema. In the following subsections we present the PExIL XML Schema organized in three groups of elements:

Textual – elements with general information about the exercise to be presented to the learner. (e.g. title, date, challenge);

Specification – elements with a set of restrictions that can be used for generating specialized resources (e.g. test cases, feedback);

Programs – elements with references to programs as external resources (e.g. solution program, correctors) and metadata about those resources (e.g. compilation, execution line, hints).
3.1 Textual elements

Textual elements contain general information about the exercise to be presented to the learner. This type of elements can be used in several phases of the programming exercise life-cycle: in the selection phase as exercise metadata to aid discoverability and to facilitate the interoperability among systems (e.g. LMS, IDE); in the presentation phase as content to be present to the learner (e.g. exercise description); in the resolution phase as skeleton code to be included in the student’s project solution.

The following table presents the textual elements of the PExIL schema and identifies the phases where they are involved.

<table>
<thead>
<tr>
<th>Table 1. Textual elements.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Element</td>
</tr>
<tr>
<td>title</td>
</tr>
<tr>
<td>creation/authors/author</td>
</tr>
<tr>
<td>creation/date</td>
</tr>
<tr>
<td>creation/purpose</td>
</tr>
<tr>
<td>challenge</td>
</tr>
<tr>
<td>context</td>
</tr>
<tr>
<td>skeleton</td>
</tr>
</tbody>
</table>

The title element represents the title of the programming exercise. This mandatory element uses the xml:lang attribute to specify the human language of the element’s content. The definition of this element in the XML Schema has the maxOccurs attribute set to unbound allowing the same information to be recorded in multiple languages. The creation element contains data on the authorship of the exercise and includes the following sub-elements: authors with information about the author(s) of the exercise organized by several author elements (represented as RDF elements\(^1\)); date which includes the date of the generation of the exercise and purpose that describes the event for which the exercise was created or the institution where the exercise will be used. The context element is an optional field used to contextualize the student with the exercise. The challenge element is the actual description of the exercise. Its content model is defined as mixed content to enable character data to appear between XHTML child-elements. This XML markup language will be used to enrich the formatting of the exercises descriptions. The skeleton element refers to a resource containing code to be included in the student’s project solution.

3.2 Specification elements

The goal of defining programming exercises as learning objects is to use them in systems supporting automatic evaluation. In order to evaluate a programming exercise the learner must submit a program in source code to an Evaluation Engine (EE) that

\(^1\) Representing vCard Objects in RDF - W3C Member Submission 20 January 2010 - http://www.w3.org/Submission/vcard-rdf/
judges it using predefined test cases - a set of input and output data. In short, the EE compiles and runs the program iteratively using the input data (standard input) and checks if the result (standard output) corresponds to the expected output. Based on these correspondences the EE returns an evaluation report with feedback.

In the PExIL schema, the input and output top-level elements are used to describe respectively the input and the output test data. These elements include three sub-elements: description, example and specification. The description element includes a brief description of the input/output data. The example element includes a predefined example of the input/output test data file. Both elements comply with the specification element that describes the structure and content of the test data.

Table 2. Specification elements.

<table>
<thead>
<tr>
<th>Element</th>
<th>Selection</th>
<th>Presentation</th>
<th>Resolution</th>
<th>Evaluation</th>
</tr>
</thead>
<tbody>
<tr>
<td>input/specification</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>output/specification</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

This definition can be used in several phases of the programming exercise life-cycle as depicted in Table 2: by 1) the content author to automatically generate an input and output test example to be included on the exercise description for presentation purposes; 2) the learner to automatically generate new test cases to validate his attempt; 3) the Evaluation Engine to evaluate a submission using the test cases.

The specification element (Fig. 1) contains two attributes and two top-level elements. The attributes line_terminator and value_separator define respectively the newline and space characters of the test data. The two top-level elements are: line which defines a test data row and repeat which defines an iteration on a set of nested elements. The number of iterations is controlled by the value of the count attribute.

The line element defines a data row. Each row contains one or more variables. A variable in the specification model must have a unique name which is used to refer
values from one or more places in the specification element. A variable is represented in the PExIL schema with the data element containing the attributes:

- **id** - defines the name of the variable. To access a variable one must use the id attribute preceded by the character $ to enable the further resolution and evaluation of XPath expressions while processing the specification model;

- **type** – defines the variable data type (e.g. integer, float, string, enum). In the case of an enumeration the values are presented as a text child node;

- **value** – represents the value to be included in the input/output test file. If filled the variable acts as a constant. Otherwise, the value can be automatically generated based on a set of constraints - the type, min, max or spec attributes;

- **min/max** – represents value constraints by defining limits on the values. The semantic of these attributes depends exclusively on the data type: may represent the ranges of a value (integer and float), the minimum/maximum number of characters (string) or a range of values to be selected from an enumeration list;

- **spec** - regular expression for generating/matching strings of text, such as particular characters, words, or patterns of characters.

The following XML excerpt shows the specification elements for the input and output test data of an exercise. The exercise challenge is *given three numbers to verify that the last number is between the first two*.

Example of the input test description: "The input begins with a single positive integer on a line by itself indicating the number of the cases following. This line is followed by a blank line, and there is also a blank line between two consecutive inputs. Each line of input contains three float numbers (num1, num2 and num3) ranging values between 0 and 1000. ".

```xml
<specification line_terminator="\n" value_separator=" ">
  <line><data id="numTestCases" type="int" value="3"/></line>
  <repeat count="$numTestCases">
    <line>
      <data id="num1" type="float" min="0" max="1000"/>
      <data id="num2" type="float" min="0" max="1000"/>
      <data id="num3" type="float" min="0" max="1000"/>
      <feedback when="$num1>$num2">
        Numbers that limit the range can be given in descending order
      </feedback>
    </line>
  </repeat>
</specification>
```

Example of the output test description: "The output must contain a boolean for each test case separated by a blank line between two consecutive outputs."

```xml
<specification line_terminator="\n" value_separator=" ">
  <repeat count="$numTestCases">
    <line><data id="result" type="enum" value="1">True False</data></line>
  </repeat>
</specification>
```
As said before, the EE is the component responsible for the assessment of an attempt to solve a particular programming exercise posted by the student. The assessment relies on predefined test cases. Whenever a test case fails a static feedback message (e.g. "Wrong Answer", "Time Limit Exceed", and "Execution Error") associated with the respective test case is generated. Beyond the static feedback of the evaluator, the PExIL schema includes a feedback element in the specification element. This element defines a dynamic feedback message to be presented to the student based on the evaluation of an XPath expression included in the when attribute. This expression can include references to input and output variables or even dependencies between both. If the expression is evaluated as true then the text child node of the feedback element is used as the feedback message.

3.3 Program elements

Program elements contain references to program source files as external resources (e.g. solution program, correctors) and metadata about those resources (e.g. compilation, execution line, hints). These resources are used mostly in the evaluation phase of the programming exercise life-cycle (Table 3) to allow the EE to produce an evaluation report of a students' attempt to solve a programming exercise.

Table 3. Program elements.

<table>
<thead>
<tr>
<th>Element</th>
<th>Selection</th>
<th>Presentation</th>
<th>Resolution</th>
<th>Evaluation</th>
</tr>
</thead>
<tbody>
<tr>
<td>solution</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>corrector</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>hints</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

A program element is defined with the programType type. This type is composed by seven attributes: id – an unique identifier for the resource; language – identifies the programming language used to code the resource (e.g. JAVA, C, C#, C++, PASCAL); compiler/executer – defines the name of the compiler/executer; version – identifies the version of the compiler; source/object - defines the name of the program source/object file; compilation – defines a command line to compile the source code; and execution– defines a command line to execute the compiled code;

There are two program elements in the PExIL schema: the solution and the corrector elements. The solution element contains a reference to the program solution file. The corrector element is optional and refers to custom programs that change the general evaluation pattern for a given exercise. The metadata about the program type resources is consolidated in the hints element aggregating a set of recommendations for the submission, compilation and execution of exercises.
4 Using PExIL

In this section we validate the PExIL definition according to: its usefulness while using the PExIL definition as input of a set of tools related to the programming exercise life-cycle (e.g. generation of a IMS CC learning object package); and its expressiveness while using the PExIL definition to capture all the constraints of a set of programming exercises in a repository (e.g. description of crimsonHex programming exercises).

4.1 Generating a IMS CC learning object package

In this subsection we validate the usefulness of the PExIL definition by detailing the generation of an IMS CC LO package based on a valid PExIL instance. An IMS CC object is a package standard that assembles educational resources and publishes them as reusable packages in any system that implements this specification (e.g. Moodle LMS).

A Generator tool (e.g. PexilUtils) uses the PExIL definition to produce a set of resources related with a programming exercise such as exercise descriptions in multiple languages or input and output test files. The LO generation is depicted in Fig. 2. The generation of a LO package is straightforward. The Generator tool uses as input a valid PExIL instance and a program solution file and generates 1) an exercise description in a given format and language, 2) a set of test cases and feedback files and 3) a valid IMS CC manifest file. Then, a validation step is performed to verify that the generated tests cases meet the specification presented on the PExIL instance and the manifest complies with the IMS CC schema. Finally, all these files are wrapped up in a ZIP file and deployed in a Learning Objects Repository. In the following sub-subsections we present with more detail these three generations.
4.1.1 Exercise description generation

For the generation of an exercise description (Fig. 3) it is important to acquire the format and the human language of the exercise description. The former is given by the Generator tool and the latter is obtained from the total number of occurrences of the xml:lang attribute in the title element of the PExIL instance.

The Generator tool receives as input a valid PExIL instance and a respective XSLT 2.0 file and uses the Saxon XSLT 2.0 processor combined with the xsl:result-document element to generate a set of .FO files corresponding to the human languages values founded in the xml:lang attribute. The following code shows an excerpt of the Pdf.xsl file. This stylesheet generates the .FO files based on the textual elements of a PExIL instance:

```xml
<xsl:template match="pexil:title">
    <xsl:variable name="uri" select="concat('desc',@xml:lang,'.fo')"/>
    <xsl:result-document href="resources/{$uri}">
        <fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
            <!-- apply templates over the textual elements --> ...
        </fo:root>
    </xsl:result-document>
</xsl:template>
```

In the next step, the .FO files are used as input to the Apache FOP formatter – an open-source and partial implementation of the W3C XSL-FO 1.0 standard - generating for each .FO file the corresponding PDF file.

Fig. 3 Generation of the exercise descriptions.

The use of the PExIL definition to generate exercise descriptions does not end here since the PExIL definition is included in the LO itself making it possible, at any time of the LO life-cycle, to regenerate the exercise description in other different formats.

The description also includes a description and an example of a test case. In the case of the absence of the input/description and input/example the Generator relies on the specification element to generate the test data and include it in the exercise description.

4.1.2 Test cases and feedback generation

The generation of test cases and feedback relies on the specification element of the PExIL definition. The Generator tool can be parameterized with a specific number
of test files to generate. Regardless of this parameter, the tool calculates the number of test cases based on the total number of variables and the number of feedback messages. In the former, the number of test cases is given by the formula $2^n$ where the base represents the number of range limits of a variable and the exponent the total number of variables. Testing the range limits of a variable is justified since their values are usually not tested by students, thus with a high risk of failure. In the latter, the tool generates a test case for each feedback message found. The generation will depend on the successful evaluation of the XPath expression included in the when attribute of the feedback element. The following example helps to understand how the Generator calculates the test cases.

```xml
<line>
  <data id="n1" type="float" min="0" max="1000"/>
  <data id="n2" type="float" min="0" max="1000"/>
  <data id="n3" type="float" min="0" max="1000"/>
  <feedback when="$num1>$num2">Numbers that ...
</feedback>
</line>
```

Suppose that the Generator tool is parameterized to generate 10 test cases. Using the previous example we can estimate the number of test cases and its respective input values as demonstrated in the Table 4.

<table>
<thead>
<tr>
<th>Var.</th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
<th>T4</th>
<th>T5</th>
<th>T6</th>
<th>T7</th>
<th>T8</th>
<th>T9</th>
<th>T10</th>
</tr>
</thead>
<tbody>
<tr>
<td>n1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1000</td>
<td>1000</td>
<td>1000</td>
<td>1000</td>
<td>Min=n2+1</td>
<td>R</td>
</tr>
<tr>
<td>n2</td>
<td>0</td>
<td>0</td>
<td>1000</td>
<td>1000</td>
<td>0</td>
<td>0</td>
<td>1000</td>
<td>1000</td>
<td>N2</td>
<td>R</td>
</tr>
<tr>
<td>n3</td>
<td>0</td>
<td>1000</td>
<td>0</td>
<td>1000</td>
<td>0</td>
<td>1000</td>
<td>0</td>
<td>1000</td>
<td>R</td>
<td>R</td>
</tr>
</tbody>
</table>

The test values are: eight tests to cover the range limits of all variables ($2^3 = 8$); one test to represent the constraint included in the feedback message. Note that this test case will be executed only if the expression included in the when attribute was not covered in the previous eight test cases; the remaining tests are generated randomly.

Also note that whoever is creating the programming exercise can statically define new test cases and use the PExIL definition for validation purposes.

### 4.1.3 Manifest generation

An IMS CC learning object assembles resources and metadata into a distribution medium, typically a file archive in ZIP format, with its content described by a manifest file named imsmanifest.xml in the root level. The main sections of the manifest are: 1) metadata which includes a description of the package, and 2) resources which contains a list of references to other resources in the archive and dependency among them. The metadata section of the IMS CC manifest comprises a hierarchy of several IEEE LOM elements organized in several categories (e.g. general, lifecycle, technical, educational). The following table presents a binding of the PExIL textual elements and the corresponding LOM elements which will be used by the Generator tool to feed the IMS CC manifest.
Table 5. Binding PExIL to IEEE LOM.

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Schema</th>
<th>Element path</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>LOM</td>
<td>lomcc:general/lomcc:title</td>
</tr>
<tr>
<td></td>
<td>PExIL</td>
<td>exercise/title</td>
</tr>
<tr>
<td>Date</td>
<td>LOM</td>
<td>lomcc:lifecycle/lomcc:contribute[lom:role='Author']/lom:date</td>
</tr>
<tr>
<td></td>
<td>PExIL</td>
<td>exercise/creation/date</td>
</tr>
<tr>
<td>Author</td>
<td>LOM</td>
<td>lomcc:lifecycle/lomcc:contribute[lom:role='Author']/lom:entity</td>
</tr>
<tr>
<td></td>
<td>PExIL</td>
<td>exercise/creation/authors/author/v:VCard/v:fn</td>
</tr>
<tr>
<td>Purpose</td>
<td>LOM</td>
<td>lomcc:general/lomcc:coverage</td>
</tr>
<tr>
<td></td>
<td>PExIL</td>
<td>exercise/creation/purpose</td>
</tr>
</tbody>
</table>

By defining this set of metadata at the LOM side, eLearning systems continue to use the metadata included in the IMS CC manifest to search for programming exercises, rather than using a specialized XML dialect such as PExIL.

4.2 Describing crimsonHex programming exercises

In this subsection we validate PExIL expressiveness by using the PExIL definition to cover the requirements (e.g. the input/output constraints of the exercise) of a subset of programming exercises from a learning objects repository.

**Evaluation of PExIL expressiveness**

![Fig. 4 Evaluation of PExIL expressiveness.](image)

For the evaluation process we randomly selected 24 programming exercises (1% of a total of 2393 exercises) from a specialized repository called crimsonHex [8]. We check manually if the PExIL definition covers all the constraints of the input/output data. The evaluation results, depicted in the Fig. 4, shows that in most cases (21 – 88%), PExIL was expressive enough to cover the constraints of the exercise test data. In just one case, we had to make a minor change in the PExIL definition to capture alternative content models.

Finally, two exercises were not completely covered by the PExIL definition. This means that using only the standard data types of PExIL we were able to define the input and output files, and these definitions can be used to validate them. However, these definitions cannot be used to generate a meaningful set of test data. In these cases the programming exercise author would have to produce test files by some other means (either by hand or using a custom made generator). In our opinion, the data
types required be these exercises are comparatively rare and do not justify their inclusion in the standard library. However, PExIL does not restrict data types and PexilUtils can be extended with generators for other data types, if this proves necessary.

5 Conclusions

In this paper we present PEXIL – a XML dialect for authoring LOs containing programming exercises. Nevertheless, the impact of PExIL is not confined to authoring since these documents are included in the LO itself and they contain data that can be used in its life-cycle, to present the exercise description in different formats, to regenerate test cases or to produce feedback to the student.

For evaluation purposes we validate the PExIL definition by using it as input for the generation of an IMS CC learning object package through a set of tools and by using it to capture all the constraints of a set of programming exercises stored in a learning objects repository called crimsonHex.

In its current status the PExIL schema is available for test and download. Our plans are to support in a near future this definition in the crimsonHex repository. We are currently finishing the development of the generator engine to produce a LO compliant with the IMS CC specification. This tool could be used as an IDE plug-in or through command line based on a valid PExIL instance and integrated in several learning scenarios where a programming exercise may fit from curricular to competitive learning.

References

2. ADL SCORM Overview. URL: http://www.adlnet.gov/Technologies/scorm.
7. IMS Common Cartridge Profile, Version 1.0 Final Specification. URL: http://www.imsglobal.org/cc/ccv1p0/imsc_profilev1p0.html.

Using Mobile Device Detection Approaches to Augment the Accuracy of Web Delivery Content

Ricardo Queirós¹ and Mário Pinto²

¹ DI-ESEIG/IPP & KMITL, Porto, Portugal
ricardo.queiros@eu.ipp.pt
² DI-ESEIG/IPP & KMITL, Porto, Portugal
mariopinto@eu.ipp.pt

Abstract. Recent studies of mobile Web trends show a continuous explosion of mobile-friendly content. However, the increasing number and heterogeneity of mobile devices poses several challenges for Web programmers who want to automatically get the delivery context and adapt the content to mobile devices. In this process, the devices’ detection phase assumes an important role where an inaccurate detection could result in a poor mobile experience for the end-user. In this paper we compare the most promising approaches for mobile device detection. Based on this study, we present an architecture for a system to detect and deliver uniform m-Learning content to students in a Higher School. We focus mainly on the devices’ capabilities repository manageable and accessible through an API. We detail the structure of the capabilities XML Schema that formalizes the data within the devices’ capabilities XML repository and the REST Web Service API for selecting the correspondent devices’ capabilities data according to a specific request. Finally, we validate our approach by presenting the access and usage statistics of the mobile web interface of the proposed system such as hits and new visitors, mobile platforms, average time on site and rejection rate.

Keywords: Device detection; XML repositories; m-learning.

1 Introduction

In a recent survey [1] at our school - ESEIG (Escola Superior de Estudos Industriais e de Gestão) - we state that a large number of our students use, on a regular basis, mobile devices. The survey shows us that they are already engaged with mobile technology and are eager to use their devices in several scenarios from accessing to the ESEIG’s web site (e.g. to consult news and events), to accessing the Learning Management System for course content, assignments and grades. Moreover, we also noticed that our students use different mobile devices with different characteristics that hinder the user mobile experience. These issues have sparked the creation of a web solution to enable the delivery of uniform web content on particular devices. This solution is composed by two sequential phases: device detection and content adaptation.
In the former, the proposed solution should determine the client characteristics and compare them with a devices’ capabilities storage system. On this phase we should obtain a fully and accurate X–ray of the client device. In the latter, the Web content (text, images, audio and video) must be selected/changed based on the characteristics previously obtained to suit the user’s computing environment and usage context.

In this paper we explore the use of device detection techniques to provide mobile users with a more rich experience. The outcomes of this study were the basis for the design of a system architecture [1] – called ESEIG Mobile –to detect and deliver uniform m-Learning content to ESEIG students. In this architecture we highlight the devices’ capabilities XML repository and the REST API Web Service. The repository aims to store a meaningful number of characteristics of mobile devices (e.g. number of colors, resolution). The REST Web Service is used to obtain these characteristics based on the client’s HTTP header request.

We validate this approach by presenting the access and usage statistics on the proposed system. This statistics were collected through the Google Analytics service, in order to better understand the adherence to a mobile web interface (e.g. average time on site, rejection rate).

The remainder of this paper is organized as follows: Section 2 enumerates and compares several technologies for the devices’ detection. In the following section we present the architecture of ESEIG Mobile and the design of its internal components. The next section we validate the ESEIG-Mobile prototype system based analyzing its usage data. Finally, we conclude with a summary of the main contributions of this work and a perspective of future research.

2 Device Detection techniques

Mobile content quality requires a full and demanding awareness of the special limitations and benefits of mobile devices [2]. Some examples of these constraints are the limited computational power, small screen size, constrained keyboard functionality and media content types supported. Due to those constraints the mobile content must be adapted to suit the mobile device characteristics. Adaptation means a process of selection, generation or modification of content (text, images, audio and video) to suit to the user’s computing environment and usage context [3]. In order to provide content adaptation, one must acknowledge the characteristics of the client device. Several approaches appeared in the last years to address this issue.

One approach is to use the common capabilities of the mobile devices and ignore the rest. Finding the Lowest Common Denominator (LCD) of the capabilities of target devices, will allow you to design a site that will work fairly well in all devices. In order to allow content providers to share a consistent view of a default mobile experience the W3C Mobile Web Best Practices Working Group (BPWG) has defined the Default Delivery Context (DDC) as a universal LCD [4]. This purpose is commonly adopt, however it limits the devices with better capabilities than LCD and decreases the use of a wider and heterogeneous mobile audience.

The most used approach is the one that obtains context information through the HTTP headers. These headers can be used to obtain the capabilities of a requesting
device such as MIME types, character sets, preferred reply encoding and natural languages. In addition to the accepted headers, the User-Agent header includes non-standard information about the device and the browser being used. This lack of standardization affects the data interpretation and extension [5]. To overcome these difficulties the device profiling concept emerged in recent years as a definition of the profile data structure that is being covered by several standards, such as CC/PP [6], User Agent PROFile (UAProf) [7] and Wireless Universal Resource FiLe (WURFL) [8].

The W3C CC/PP specification defines how client devices express their capabilities and preferences (the user agent profile) to the server that originates content (the origin server).

The UAProf is a standard created by the Open Mobile Alliance (formerly the WAP Forum) to represent a concrete CC/PP vocabulary for mobile phones and defines an effective transmission of the CC/PP descriptions over wireless networks. Mobile phones that are conformant with the UAProf specification provide CC/PP descriptions of their capabilities to servers that use this information to optimize the content. The information is formatted in XML containing several attributes (e.g. screen size, color and audio capabilities, operating system and browser info, encoding).

WURFL is a repository describing the capabilities of mobile devices worldwide. It uses an XML configuration file which contains a comprehensive list of device capabilities and features. A huge community of developers contributes with device information feeding the WURFL file and reflecting the new mobile devices coming on the market. Nowadays, WURFL shares the hegemony on the device detection market with other products such as DeviceAtlas [9] and Mobile Device Detect [10].

DeviceAtlas is a commercial database for device detection created by dotMobi. DeviceAtlas incorporates many device databases and sources such as WURFL and UAProf and retrieve accurate JSON results. Recently, the project was updated by the DeviceAtlas Personal - a SOA aware version. It works as follows: a user visits a Web site on his mobile device. Then the server forwards the User-Agent HTTP request header to the DA Personal service, and receives a response containing information about the user's device.

The Mobile Device Detect (MDD) project is a PHP solution for device detection. It is free for non-commercial sites. Rather than using a comprehensive user agent database, this project is based on a script that seeks for specific string fragments in the user agent string.

The following table presents a mobile device concurrency test [11].

Table 1. Mobile device concurrency test.

<table>
<thead>
<tr>
<th>Method</th>
<th>Time (seconds)</th>
<th>Mobile</th>
<th>Non-Mobile</th>
</tr>
</thead>
<tbody>
<tr>
<td>WURFL API</td>
<td>20.8</td>
<td>1090</td>
<td>482</td>
</tr>
<tr>
<td>DeviceAtlas API</td>
<td>1.2</td>
<td>527</td>
<td>1045</td>
</tr>
<tr>
<td>Mobile Device Detect</td>
<td>1.3</td>
<td>684</td>
<td>888</td>
</tr>
</tbody>
</table>

The data set includes 1,572 unique user agents. We can say that accuracy and performance are the two most important features to take into consideration when
selecting a device detection mechanism. Device detection is not guaranteed to be 100% accurate since user agent strings are highly variable and non-standardized. At the same time, DeviceAtlas and MDD present smaller processing times, but more inaccurate results than WURFL.

3 ESEIG-Mobile

In this section we present the architecture of ESEIG-Mobile as a new layer on the top of the existent ESEIG infrastructure. This project aims to standardize the delivery of learning content produced at our School (ESEIG) to the diversity of mobile devices used by our students. In the following subsections we present the overall architecture of the ESEIG-Mobile and its main components.

3.1 Architecture

The architecture of the ESEIG-Mobile system is described by the component diagram shown in Figure 1.

![Component’s diagram of the ESEIG-Mobile system.](image)

A typical execution flow will be as follows: the client device makes (1) an HTTP request; the Detector component at server-side invokes (2) a REST service with the user agent as parameter (collected in the HTTP user-agent header of the client request). The service seeks on the WURFL database and returns the respective capabilities to the Adapter component. The Adapter component based on the previously achieved characteristics of the device interacts (3) with the Connector component to select the more suitable content to compose the HTTP response back to the client.
3.2 Devices’ Capabilities Repository

The Devices’ Capabilities Repository contains a file with a large list of device features based on WURFL. WURFL is an open source database (file called wurfl.xml) of wireless device capabilities. The WURFL structure is currently formalized in a Document Type Definition (DTD) file. The following figure shows an overall view of the respective WURFL XML Schema.

Fig. 2. The WURFL schema.

The schema has two top-level elements: the version and the devices elements. The version element is composed by a set of sub-elements: ver – the version of the WURFL database; last_update – the date of the last update of the database; maintainers – a set of maintainer elements related with the person(s) responsible by maintaining the database; authors – a set of author elements related with the person(s) responsible by creating the database;

The devices element contains one or more device sub-elements that model a certain device. This element contains the user_agent attribute, the device id attribute (created by the WURFL maintainer), the fall_back attribute (gives a way to infer more information about the device) and the actual_device_root attribute to signal that the current device element may be chosen as the representative for all devices by the same brand and model name.

In addition to this data, a device element may carry information about device features commonly referred to as capabilities. A device capability is an XML fragment which contains information about a specific feature of a given device. The device capabilities are organized in groups. Groups are used to improve the readability of the WURFL XML database by humans. For instance, Nokia phones support tables because fall_back is defined as generic (WURFL default) as described in the following piece of code.

```xml
<device user_agent="Nokia" fall_back="generic"
        id="nokia_generic">
  <group id="ui">
    <capability
      name="break_list_of_links_with_br_element_recommended"
      value="false" />
  </group>
</device>
```

The WURFL is based on the concept of family of devices. All devices are descendent of a generic device, but they may also descend from more specialized families. This mechanism, called 'fall_back', lets programmers derive the capabilities of a given
phone by looking at the capabilities of its family, unless a certain feature is specifically different for that phone [8].

The WURFL repository can be either installed locally and be synchronized with a WURFL public repository where the developers’ community makes updates regularly or accessed remotely through the use of a REST Web Service.

### 3.3 REST Web Service

A Web browser, when requesting a web page, sends a set of HTTP headers to the server. One of these headers is the User-Agent header that contains information about the user agent originating the request. The field can contain multiple product tokens and comments identifying the agent and any sub-products which form a significant part of the user agent as stated in the RFC 2616 [12]:

\[
\text{User-Agent} = "\text{User-Agent}" : I*( \text{product | comment} )
\]

For instance, an Android mobile device may send the following user agent string:

\[
\text{User-Agent: Mozilla/5.0 (Linux; U; Android 2.2; pt-pt; GT-I9000 Build/FROYO) AppleWebKit/533.1 (KHTML, like Gecko) Version/4.0 Mobile Safari/533.1}
\]

The Detector component receives and uses it to query the WURFL device repository through the WURFL Web Service. This service provides a RESTful interface to the WURFL database. The use of this approach rather than a local implementation of WURFL avoids the maintenance of a local storage liable to the typical synchronization issues.

The following table details the WURFL API interface.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>ua</td>
<td>User-Agent</td>
<td><a href="http://api.wurflws.com/wurflws?ua=%5BUA">http://api.wurflws.com/wurflws?ua=[UA</a>]</td>
</tr>
<tr>
<td>search</td>
<td>Capabilities filter</td>
<td><a href="http://api.wurflws.com/wurflws?search=%5BF1">http://api.wurflws.com/wurflws?search=[F1</a></td>
</tr>
</tbody>
</table>

The API’s endpoint is http://api.wurflws.com/wurflws. The API has two parameters: ua and search. The ua parameter defines the User-Agent string that identifies the device. If not sent then the original User-Agent header is used to find the corresponding device. The search parameter represents the Capabilities filter. Only these capabilities (is_wireless_device, brand_name, model_name, resolution_width, resolution_height, full_flash_support, flash_lite_version, mobile_browser, device_os, ajax_xhr_type, ajax_support_javascript) should be returned if these parameters are sent. The capabilities should be separated by a pipe.

For instance, calling the following URL will return the capabilities of the Nokia 6630 with two filtered capabilities: resolution width and height:
Validation

In this section we validate the usage of the ESEIG-Mobile web interface, characterizing the access and rejection levels based on Google Analytics service data, such as hit counters, rejected requests, new visitors, traffic and mobile operating systems used to access the ESEIG-Mobile interface. The data was collected from November 2010 to February 2011.

Regarding the access rate (Figure 3), one can consider that although the access rate is relatively low, it has increased significantly. The amounts collected can result from the fact that the platform is very recent, and therefore still unknown by most students. Moreover, the high rates of new visitors may indicate that the ESEIG-Mobile web interface starts to be increasingly popular. This is reinforced by the high rate of new visitors, always above the 60%. This clearly shows that the service is gradually being known by students and teachers.

Figure 4 shows the average time spent on the ESEIG-Mobile web interface and the number of pages visited (average) by access. This data are useful to evaluate the degree of interaction of each user and how it relates with the mobile platform. In fact, the average time on site is between one and two minutes, and the number of pages visited around two, by access. This data is in compliance with that obtained in ESEIG-desktop web interface, also collected through Google Analytics in the same period of time: average time on site around two minutes and number of pages retrieved 2.5. This is an interesting issue, since it shows that users remain interested...
with the contents offered, at least to a similar degree to what happens with the desktop web interface. Other important conclusion is related with the average time on site that lies between 1 and 2 minutes. This fact indicates that the accuracy of the detector component is acceptable otherwise an incorrect approach would considerably decrease the current value.

![ESEIG Mobile users activity](image)

**Fig. 4.** ESEIG-Mobile activity on the site.

Another important issue that arises from the data analysis is the diversity and heterogeneity of the client devices. Symbian, Android, iPhone and iPad are the leading mobile devices, but there is a large number of other devices that ESEIG-Mobile system should respond. The support offered for different platforms and mobile operating systems is, certainly, a critical success factor for the ESEIG-Mobile web interface.

Figure 5 illustrates each of the mobile access platforms used by students, as well as its incidence. This in an important data, since it allows us to understand which are the mobile platforms most commonly used, and it returns some important feedback regarding the efficiency and effectiveness of the proposed approach detailed in this paper. In fact, Symbian and Android are the main platforms used to access the ESEIG-Mobile interface, followed by iPhone, iPad and iPod. A surprising fact is the lower number of devices with the Windows Mobile operating system.
5 Conclusions

In this paper, we present and compare several approaches for defining delivery context. Based on a previous survey and aided by this comparative study we present the design of ESEIG-Mobile - an open system for the delivery of suitable and uniform e-Learning content to the mobile devices of ESEIG students. The ESEIG-Mobile system relies on a devices’ capabilities repository to store a meaningful number of characteristics of mobile devices and on a REST Web Service to obtain these characteristics based on the client’s HTTP header request.

To validate our approach we present the access and usage statistics of the ESEIG-Mobile project based on the Google Analytics data. The analysis of this data is very important since it helps us to confirm and understand the heterogeneity of the students’ mobile devices and their usage habits and preferences. It also helps to identify and find the best approaches to improve the ESEIG-Mobile system.

In this moment ESEIG-Mobile is in early development as we are only detecting if the HTTP request is made from a mobile device and query some device capabilities from the WURFL device repository. We expect some challenges in the prototype implementation process regarding, for instance, the transformation of the Web resources in the WNG format [13]. For this task we are considering using Extensible Stylesheet Language for Transformation (XSLT) to formally describe the transformations. Other ongoing work is related with increasing the device repository performance migrating from the WURFL XML database to a relational database (e.g. MySQL) using the Tera-WURFL project [14].
References


XML Schemas for Parallel Corpora

Alberto Simões and Sara Fernandes

1 Centro de Estudos Humanísticos, Universidade do Minho, ambs@ilch.uminho.pt
2 Departamento de Informática, Univ. do Minho, sara.fernandes@di.uminho.pt

Abstract. Parallel corpora are resources used in Natural Language Processing and Computational Linguistics. They are defined as a set of texts, in different languages, that are translations of each other. Note that these translations do not need to cover the full document, as we might have sentences translated just on some of the languages.

When dealing with the process of sharing resources, recent years have bet on the use of XML formats. This is no different when talking about parallel corpora sharing. When visiting different projects in the web that release parallel corpora for download, we can find at least three different formats. In fact, this abundance of formats has led some projects to adopt all the three formats.

This article discusses these three main formats: XML Corpus Encoding Standard, Translation Memory Exchange format and the Text Encoding Initiative. We will compare their formal definition and their XML schema.

1 Introduction

Natural Language Processing and Computational Linguistics are examples of areas where corpora and, in particular, parallel corpora, are relevant resources. To best understand the concepts we will discuss, we should start by defining this concept.

The corpus (plural, corpora) term, born in Linguistics, refers to a finite collection of texts, usually from a restricted domain [5]. There are hundreds of examples of available corpora. The most well known is the British National Corpus[3].

A Parallel Corpus is a collection of texts in different languages, where each of them is a translation of each other. In some situations one of these languages is considered as the source language, and its translations as the target languages. While not consensual, it is usual to consider that a parallel corpus is aligned at the sentence level, meaning that there is a relationship between sentences (or, roughly, text sequences) in the different languages.

This alignment process is defined as: having two parallel texts, $U$ and $V$, a sentence alignment of these texts is a segmentation of $U$ and $V$ in $n$ segments, such that, for each $i$, $1 \leq i \leq n$, $u_i$ and $v_i$ are mutual translations, and $u_i$ and $v_i$ are, respectively, sequences of sentences from $U$ and $V$ [4].

Note that this definition means that we might have segmentations $u_i$ or $v_i$ that are empty sequences from $U$ and $V$. Therefore, there might exist sentences in one of the languages that does not have a corresponding translation. Indeed, the creation or removal of sentences during the translation process is common.

This definition can be expanded to a set of languages, instead of just a pair. In this situation, we have a set $S$ of $m$ texts $T_i (1 \leq i \leq m)$, that have $n$ segments each, such that, $\forall i, j \ 1 \leq i \leq m \land 1 \leq j \leq n$, $t_{i,j}$ sequences of sentences are mutual translations.

The Parallel Corpora definition is nothing more than this mapping between segments in different languages. Researchers, being in the field of Natural Language Processing or Linguistics, like to enrich their parallel corpora with extra information. The kind of information to be added will highly depend on the corpus objective. Examples encompass the simple annotation of named entities (personal or company names, for instance), morphologic or part-of-speech tagging of each word, syntactic structure, etc.

This diversity of possible annotations makes it almost impossible to define a standard schema with all the alternatives one might want. Therefore, the adopted solution is the ability to define generic tags that each user can personalize.

In this article we will focus on three different formats that have been used by the research community to encode parallel corpora:

- The Text Encoding Initiative (TEI) schema (subsection 2.1);
- The Translation Memory Exchange (TMX) schema (subsection 2.2);
- The XML Corpus Encoding Standard (XCES) schema (subsection 2.3);

In the next section we will explain where they came from and the original purpose for which they were created. Their objectives are very different, which means that the level and type of annotation they can support is diverse. Nevertheless, they can all encode non-annotated parallel corpora, meaning it should be possible to define computational bridges to convert between these formats.

While section 2 will present each of these formats in particular, section 3 will compare their structure in means of usability and flexibility. Finally, section 4 discusses the directions users who need to encode parallel corpora should follow.

2 Parallel Corpora Encoding Standards

This section title is misleading, as just one of the formats (XCES, section 2.3) was developed specifically for XML corpora encoding.

All the formats we will discuss are currently being used by researchers to release parallel corpora and, some of these researchers, are making their corpora available in more than one format.

In this section we will not compare the schemas but, instead, define the subset that are relevant to encode parallel corpora and annotate possible language phenomena. Finally, we will perform a qualitative evaluation on their flexibility to encode parallel corpora (check section 3).
The Text Encoding Initiative (TEI) collection of schemas [8] was created to help in preparation and interchange of electronic texts for most real-world situations. TEI is modular, and depending on the text being encoded the set of schemas to be used is different. TEI includes a big variety of schemas, to encode texts, verses, transcription of speech, standard dictionaries, lists of places and names (toponyms and onomastic indexes), tables, mathematical formulae, graphs, networks, trees and others.

In particular, TEI includes schemas to encode language corpora (chapter 15 of the TEI Guidelines for Electronic Text Encoding and Interchange) and for text segmentation and alignment (chapter 16).

All these schemas share a common schema, known as the TEI header. This header includes typical meta-information, as the name of the document, its authors, the document copyright, editor, publisher, year, etc. While meta-information is relevant when encoding corpora and parallel corpora, in this article we will be more interested in the means these schema have to encode the corpora, itself.

Nevertheless, we should stress the relevance of meta-information for corpora construction. It is very relevant to know the genre of the text (journalistic, literary, religious, etc), the age of the text (when it was written), its language and sub-languages, its type (oral, written), etc. All this information can be stored in the TEI header.

The macro-structure of a TEI corpus can be described as follows:

\[
\text{teiCorpus} \leftarrow \text{teiHeader}, (\text{TEI} \mid \text{teiCorpus})^+ \\
\text{TEI} \leftarrow \text{teiHeader}, \text{text} \\
\text{text} \leftarrow \text{front} ?, (\text{body} \mid \text{group}), \text{back} ? \\
\text{group} \leftarrow (\text{text} \mid \text{group})^+ 
\]

Note that this structure is quite rich. It is possible to have a header for the full corpora, and a separated header for each text. Also, each text might be grouped in different sections.

The text element is used by TEI to store all kind of texts. Therefore one can expect all kinds of mark-up to be possible inside this element. Although there are some corpus that might come from well structured data sources, most are processed by automatic tools, that just extract pure text. Therefore we can consider that a text is just a sequence of paragraphs (\text{p} element) or lines (\text{l} element, often used for verse lines).

Some texts include some other level of segmentation, like the \text{div} element, that is used to divide text into sections.

For text annotation, TEI provides elements below the line or paragraph level. It includes elements for sentences (\text{s} element), for clauses (\text{cl} element), phrases (\text{phr} element) and words (\text{w} element). In fact it provides elements below word level, as morpheme, character or punctuation character.

Given the amount of elements to annotate different levels of text, the annotation of a corpus in TEI format can be very detailed. Any one of these elements
can have attributes like type and function for phrases and clauses, lemma and type for words. Therefore, it is very simple to add all the needed information with these attributes, that have an open content type.

As for the alignment task, it is implemented as links between elements. Usually (but not necessarily), parallel corpora are encoded in TEI as three separate files: the text in the source language, the text in the target language, and the alignment file. This alignment file includes the usual TEI header, and a sequence of linkGrp elements. These elements have some meta-information, like the documents that are being linked (in the xtargets attribute), and includes a list of link elements. These elements can include a type attribute (that is usually the number of segments form the source-text and from the target-text that are being linked), and a xtargets or targets attribute that has the identifiers used in the individual text files for the p or l elements (although this mechanism makes it easy to link sub-paragraph parts, like sentences, clauses, phrases or even words).

As an example for a linkGrp element:

```xml
<linkGrp targType="head p" xtargets="jrc-pt;jrc-ro">
  <link type="1-1" xtargets="28;28"/>
  <link type="1-1" xtargets="30;30"/>
  <link type="1-1" xtargets="31;31"/>
  <link type="1-2" xtargets="32;32 33"/>
  <link type="1-2" xtargets="33;34 35"/>
</linkGrp>
```

More than two languages support is easy to perform, extending this mechanism. In fact, we can find two different solutions: first, instead of two text files, we have one per language, and instead of a linkGrp, we have a set of groups, one for each language pair; other solution is to have more than two fields in the targets or xtargets attributes.

This description on the TEI mechanisms for encoding corpora and their alignment wasn’t very detailed as we do not intend to write a tutorial, but instead, to compare the formats. Therefore, we invite the interested reader to consult the Guidelines for Electronic Text Encoding and Interchange that are available on the web.

TEI is a very detailed schema. Therefore, there is mostly any kind of text that can not be encoded as a TEI XML file. The drawback is the leaning learning curve.

As an example of project/corpus encoded in TEI, please check the multilingual parallel corpus based on the Acquis Communautaire, the total body of European Union (EU) law applicable the the EU Member States [6].

---


5 [http://wt.jrc.it/lt/Acquis/](http://wt.jrc.it/lt/Acquis/)
2.2 TMX: Translation Memory Exchange

The Translation Memory Exchange format was designed for the interchange of translation memories across different vendors of computer assisted translation (CAT) software. It is a standard, or norm, defined by the Localisation Industry Standard Association\(^6\) (LISA). LISA is an association where some Universities and the major companies with CAT software or localization offices have a seat. Examples of partners are Abbyy, Adobe Systems, Autodesk, Cisco Systems, Dell, Hewlett-Packard, ICANN, Intel Corporation, Lucent Technologies, OASIS, SDL International, Skype, Trend Micro, VMWare and XEROX.

To understand the idea of translation memory it is helpful to explain how a CAT software works. When performing a translation task, the translator is faced with sentences already translated by herself or by someone on her group. Therefore, a CAT tool stores in a database all performed translations. These translations are stored sentence by sentence (or sequence of words by sequence of words, since the reuse of translations is more effective with short sequences of words).

Therefore, a translation memory can, in a simplified way, be seen as a set of pairs that relate sequences of words in two different languages. This informal definition is quite near the definition of parallel corpora. Note that for parallel corpora we are forcing an order, a thing that translation memories do not guarantee by themselves. Given that translation memories are stored in XML files an implicit order (the order of appearance) exists. This makes the TMX format relevant for storing parallel corpora.

There is a working draft on TMX version 2.0, dated of 2007. Unfortunately no developments have been done on this proposal. Therefore all CAT tools and researchers using TMX are using the 1.4b specification.

Again, please be aware that we are simplifying the structure of TMX removing elements not relevant for the purpose discussed here. The macro-structure of a TMX file is defined as follows\(^7\):

\[
\begin{align*}
tmx & \leftarrow \text{header, body} \\
\text{header} & \leftarrow \text{@creationtool, @segtype, @srclang, @adminlang,} \\
& \quad (\text{note} \mid \text{prop})^* \\
\text{note} & \leftarrow \#PCDATA \\
\text{prop} & \leftarrow \text{@type, #PCDATA} \\
\text{body} & \leftarrow \text{tu}^* \\
\text{tu} & \leftarrow \text{@srclang, @segtype, ((note \mid prop)^*, tuv^*)} \\
\text{tuv} & \leftarrow \text{@xml : lang, ((note \mid prop)^*, seg)} \\
\text{seg} & \leftarrow \#PCDATA
\end{align*}
\]

\(^6\) http://www.lisa.org/

\(^7\) Attributes are denoted with the @ symbol. Also, the seg element definition is simplified.
Explaining, a TMX file is a header with some meta-information and a body with a sequence of translation units \((tu)\). A translation unit is a sequence of translation unit variants \((tuv)\) with a segment \((seg)\). Figure 1 presents a simple TMX file.

```xml
<?xml version="1.0"?>
<tmx version="1.4">
  <header creationtool="XYZTool" creationtoolversion="1.01-023"'
datatype="PlainText" segtype="sentence"'
  adminlang="en-us" srclang="EN" />
  <body>
    <tu>
      <tuv xml:lang="en"><seg>hello</seg></tuv>
      <tuv xml:lang="it"><seg>ciao</seg></tuv>
      <tuv xml:lang="pt"><seg>olá</seg></tuv>
    </tu>
    <tu>
      <tuv xml:lang="en"><seg>world</seg></tuv>
      <tuv xml:lang="en"><seg>earth</seg></tuv>
      <tuv xml:lang="it"><seg>mondo</seg></tuv>
      <tuv xml:lang="pt"><seg>mundo</seg></tuv>
    </tu>
  </body>
</tmx>
```

**Fig. 1.** Example of a simple TMX file.

Meta-information can be added at different levels. As the \textit{prop} and \textit{note} elements are open content they can be used mostly for everything. Also, as they can be added at different levels (\textit{header}, \textit{tuv} or \textit{tu}) they make it easy to annotate specific translation units or units variants. Unfortunately there is not a way to aggregate translation units in blocks. This is a problem if you wish to tag each translation unit with the source where the text came from. With TMX we have only two options: create a different TMX for each text source or to tag each translation unit with the text source. If we had a way to create blocks we could associate that information to blocks.

Regarding word annotation, TMX files support is very poor or inexistent. It supports some in-line tags but only one can be barely used to annotate text. Its name is \textit{hi}, standing for \textit{highlight}, and has only two possible attributes: \textit{type} and \textit{x}. The first is for free use (and therefore the user can invent their own way to encode any desired information), and the second is used to match elements between translation units. That is, lets the user link words or segments between translations.
Of course one can add a namespace to the XML file to perform the annotation. In this article we are interested only on the native mechanisms each of these formats provide to the user.

The TMX file format is also being used to make available parallel corpora. As an example, check the OPUS\(^8\) project [7], that includes different types of corpora to download in TMX format.

### 2.3 XCES: XML Corpora Encoding Standard

The XCES (XML Corpora Encoding Initiative) encoding specifications has been developed for and by the language engineering community, with the aim to provide guidelines for encoding various features in written text, morphosyntactic annotation, and alignment information, all of which are relatively stable and agreed-upon within the community.

XCES\(^9\) is the instantiation of Corpus Encoding Standard (CES\(^10\)) as an XML document. CES was developed when SGML (Standard Generalized Mark-up Language) was broadly used, which explains CES not being originally developed in XML. One of the main problems of XCES is being based on CES. Authors did not write documentation on XCES relying on CES documentation. Unfortunately, portions of the standard were changed and, based in the well known Murphy’s Law, the way to encode alignments in XCES changed.

It follows the same concept of TEI. Instead of defining a single schema for encoding corpora, it defines a family of smaller schemas that can be combined together to achieve different kinds of annotation, accordingly with the user needs. This allows more flexibility on their use.

In this article we will look specifically to the schema designed to encode parallel corpora.

A formal view of the macro-structure of a XCES alignment document follows:

```
cesAlign ← @fromDoc, @toDoc, @type, cesHeader, linkList
linkList ← linkGrp\(^+\)
linkGrp ← @fromDoc, @toDoc, @type, link\(^+\)
link ← align\(^+\)
```

That is, an alignment document in XCES is divided in two main sections, just like most standards, an header with meta-data (that we will not dissect in this paper) and a body, named linkList where the relations between segments will be defined.

This linkList is usually divided in linkGrp, which are groups of alignments for a specific file. So, if our alignment document is specifying alignments among more than one pair of files, then the alignment document will have a linkGrp element for each document pair.

---

\(^8\) [http://opus.lingfil.uu.se/index.php](http://opus.lingfil.uu.se/index.php)
\(^9\) [http://www.xces.org/](http://www.xces.org/)
\(^10\) [http://www.cs.vassar.edu/CES/](http://www.cs.vassar.edu/CES/)
This schema should be possible to use in cases when only one pair of documents is being aligned (therefore with just one linkGrp) or cases when more than one pair are being aligned. The schema supports the attributes fromDoc, toDoc and type at two different levels, which makes it possible to define these attributes at the top level of the document, at the cesAlign element, emphasizing this information. The fromDoc and toDoc attributes are simple URI that point to the files being aligned. The type attribute specifies the type of alignment (paragraph, sentence, word).

Inside the linkGrp element, we will have each alignment information, in link elements. Unfortunately the documentation is missing, and the authors are not answering e-mail. This leads to a problem: the user needs to guess the semantic of the XML structure defined by the Schema.

The link element includes a sequence of align empty elements. The pointer from each align element to the text being aligned is performed using an href attribute. But no further information on how to fill in this element is given to the user. Also, given that linkGrp elements just have information to a pair of documents, it is quite strange that the link elements support more than two align elements.

Regarding the annotation of the documents, XCES has a detailed schema to annotate the documents structure. In fact, and although it is not as detailed as TEI, it includes a very good set of entities to encode paragraphs, lists, tables, images, poems, etc.

Finally, the word level annotation is obtained with yet another XCES schema. Unfortunately, this schema cannot be merged with the document annotation schema. Note that unlike TEI, where each schema can be imported in top of each another, as they all share the same root structure (you can see it as a super class, TEI base, and a set of instances, one for each type of document), XCES defines a complete new schema for each kind of information (document structure, alignment, and now word level annotation). They only share a header, where the meta-information can be added.

The main problem is that an word level annotation file (or, as XCES calls it, a chunk sequence file) is just a XCES header and a sequence of chunks. These chunks have linking information where the annotation can be aligned with the document itself (so, the document is stored in a file, the annotation in another file that includes information about what portions of the file are being annotated).

Each of these chunks, include a sequence of analysis (called feat in XCES documentation, probably as an abbreviation to feature and not the English word). These elements are a key/value pair, where the user can include the type of information he would like.

The main advantage of this approach is flexibility. The user can encode virtually anything, but it is not easy to maintain. Consider the annotation of part-of-speech for each word in a text (say, the type—verb, adverb, adjective, etc.—, genre, number and verbal tense). For each one of these properties a feat element will be needed. And for each word, a chunk element with the proper linking information will be required. This is totally inefficient for processing purposes.
3 Comparing TEI, TMX and XCES

As described in the previous section, these three formats are quite different, and they were designed for different objectives. Table 1 compares some of the most relevant features of these formats. Note that we are comparing them with parallel corpora encoding in mind. So, documentation refers to the documentation on how to use these formats to encode parallel corpora, and dedicated tools, the availability of tools to encode and manage parallel corpora using these formats.

<table>
<thead>
<tr>
<th>Feature</th>
<th>TEI</th>
<th>TMX</th>
<th>XCES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Documentation</td>
<td>++</td>
<td>+</td>
<td>–</td>
</tr>
<tr>
<td>Schema simplicity to encode parallel corpora</td>
<td>–</td>
<td>++</td>
<td>–</td>
</tr>
<tr>
<td>Multi-language support</td>
<td>++</td>
<td>++</td>
<td>+</td>
</tr>
<tr>
<td>Sentence level alignment meta-data</td>
<td>–</td>
<td>++</td>
<td>–</td>
</tr>
<tr>
<td>Word level annotation</td>
<td>++</td>
<td>□</td>
<td>++</td>
</tr>
<tr>
<td>Dedicated tools</td>
<td>–</td>
<td>++</td>
<td>–</td>
</tr>
<tr>
<td>Availability of encoded corpora</td>
<td>+</td>
<td>++</td>
<td>–</td>
</tr>
</tbody>
</table>

Table 1. TEI, TMX and XCES comparison table (++ stands for pretty good, + for enough, and – for limited support. A □ is used when no support is present).

A final decision on what encoding schema to use will highly depend on your objectives. Some examples and decisions you might take:

- Your parallel corpora will be used as a translation memory for machine translation software. In this case, it is clear that TMX format should be chosen;
- You have a bunch of XML files that you would like to align at sentence level. In this situation, using TEI or XCES would be better suited, as you can just create independent alignment files that will retrieve the parts being aligned from the independent XML files.
- You are making available a multi-language corpora, in alignment pairs. Then, it is easier to release each language as a separate XML file, and independent alignment files for each language pair. This way, the user can clearly choose what file to download.

The decision will also be highly dependent on what tools are available to manage your files. As it is described in the table above, TMX is well served with tools to manipulate translation memories. From a wide range of computer assisted translation tools, to small GUI tools or even libraries, like XML::TMX [1]. TEI is quite served on tools when used as a schema to encode textual document. To manipulate parallel corpora there are just some few scripts developed by researchers that release their corpora in TEI format. Finally, XCES have been quite neglected in the last years. For example, the OPUS project, already mentioned, is trying to encode their texts in XCES. But they are following the CES documentation and using XML format. The lack of proper documentation is making this standard completely unusable.
4 Conclusions

In this article we gave a brief insight of the three major schemas available to encode parallel corpora. As the previous section showed, if we compare directly the features for each standard, we will end up selecting TEI as the best. It is not just well documented but it also includes in-depth discussion on the schema features. The biggest drawback is related to its embracing philosophy. As all kind of texts can be encoded in TEI it makes it quite difficult to develop robust tools that can handle the full schema.

The TMX format is in the other end of the continuum. It was developed for a specific purpose, it is very simple and fully functional for its main objectives. Being small, makes it quite easy to develop tools manipulating it\textsuperscript{11}; all computer aided translation software have import/export facilities for this format.

XCES is in the middle. Is was designed for a specific purpose, but generic enough to embrace a bigger set of documents related with that purpose. Its main problems are related to the lack of documentation and lack of usage. In fact, some researchers claim they are releasing their corpora in XCES format, but they are just encoding CES in XML, and XCES is more than that.

How to chose one of them is a problem. But for sure, the authors do not recommend XCES. It lacks documentation, it is not implemented on any tool, no project adopted it and, more important, the authors are working on some other standard (GRaF [3]) and are not maintaining XCES anymore.

The biggest conclusion we can get from the analysis of these three standards is that the fact of a specific standard being developed and thought for a specific type of usage it does not mean that researchers will adopt it. There are two main details that are crucial for the community to adopt a specific schema:

- If it is somewhat complicated, it should be very well documented. If it is more simplistic, some lighter documentation should be enough. But, without any kind of documentation it is hard for any researcher to give credit to that schema.
- If the schema was defined by more than one person, and in special, was defined by teams of well known departments, it should mean that these teams are interested on it. Therefore, some results, comprising results and/or tools, should be available. These tools/results should be relevant enough to convince researchers to look to that specific schema.
- To define a proper XML schema is not enough to know the field that is being annotated. A proper formation on mark-up languages is indispensable.

Authors are convinced that these factors are the main factors for the current status of XCES.

In the Per-Fide project [2] one of the main goals was to make available all the constructed corpora in the three formats: XCES, TMX and TEI. After this analysis, the authors are targeting their tools only on TEI and TMX formats.

\textsuperscript{11} In fact only 90% of the schema is really used on most tools, but this subset includes the most relevant features.
Acknowledgments

This work was funded by the project Português em paralelo com seis línguas (Português, Español, Russian, Français, Italiano, Deutsch, English) grant PTDC/CLE-LLI/108948/2008 from Fundação para a Ciência e a Tecnologia.

References

XCentric-based Visual Approach to Web Content Verification

Liliana Alexandre\(^1\) and Jorge Coelho\(^2\)

\(^1\) Lusitânia - Companhia de Seguros, S.A.
liliana.alexandre@lusitania.pt

\(^2\) School of Engineering of the Polytechnic of Porto (ISEP) & Artificial Intelligence and Computer Science Laboratory of the University of Porto (LIACC)
jam@isep.ipp.pt

Abstract. In this paper we present a tool for visually imposing constraints over the content of XCentric-based webpages and automatically repair such webpages in case they don’t comply with the imposed constraints. The tool is based in the XCentric programming language and relies on a highly declarative model.

1 Introduction

VeriFLog \([7]\) is an extension of the XCentric language \([9]\) for semantic verification of XML-based content. It relies on the unification with terms with flexible arity symbols and sequence variables which enables a compact description of constraints. It also adds builtins to enhance the development of programs in the content verification domain. The main drawback of VeriFLog is that the user needs to have at least some basic knowledge of Logic Programming in order to use it. The tool presented here enhances VeriFLog by capturing the core features and adding new ones in a user-friendly visual approach which reduces the need of previous knowledge of Logic Programming. The main application of this tool is to verify content on collaborative websites such as Wikipedia \([11]\).

The remaining of this paper is organized as follows, in section 2 we explain briefly the main concepts behind the XCentric language and the VeriFLog tool. Then, in section 3 we show how to compose rules for verifying XML-based webpages using our visual approach. In section 4 we present the related work and finally in section 5 we conclude and present future work.

We assume that the reader is familiar with Logic Programming \([14]\).

2 Verifying XML Content

Here we explain how to verify content in webpages by using the XCentric language \([9]\) and VeriFLog \([7]\).

\* Partially funded by LIACC through Programa de Financiamento Plurianual of the Fundação para a Ciência e Tecnologia (FCT)
2.1 XCentric

XCentric extends Prolog with terms with flexible arity symbols and sequence variables. This approach enables a highly declarative model for querying content in webpages. Constraints of the form \( t_1 = * = t_2 \) are solved by a non-standard unification that calculates the corresponding minimal complete set of unifiers. Details about the implementation of this non-standard unification can be found in [6]. In XCentric an XML document is translated to a term with flexible arity function symbol. This term has a main functor (the root tag) and zero or more arguments. Although XCentric translates attributes to a list of pairs, we will omit them for the sake of simplicity. Consider the simple XML file presented bellow:

```
<addressbook>
  <record>
    <name>John</name>
    <address>New York</address>
    <email>john.ny@mailserver.com</email>
  </record>
  ...
</addressbook>
```

The equivalent term is:

```
addressbook(record(name('John'),
    address('New York'),
    email('john.ny@mailserver.com'))
  ,...)
```

If we want to get the names of the people living in New York and assuming that the document is stored in variable \( Doc \) we can simply solve the following constraint:

\[
Doc = * = addressbook(\_, record(name(N), address('New York'), \_), \_).
\]

All the solutions can then be found by backtracking (in variable \( N \)).

Note that \( \_ \) is an unnamed sequence variable which unifies with any sequence. So, no matter how many records the address book has, we can describe our constraint in a very compact way by focusing on the ones that matter for our purposes. The details of the language and several illustrating examples can be found in [9]. Although the operator \( = * = \) supports variables in both sides we implemented a version which supports variables only on the right-hand side (operator \( =~\_ = \)). This is enough for processing and querying documents (which don’t have any variable inside) and increases performance. So, in the previous example, since \( Doc \) is an XML document without any variables, the operator \( =~\_ = \) could be used, giving the same results. In the tool we describe in this paper we only use operator \( =\_~\_ = \).
2.2 VeriFLog

In [7] and [8] XCentric was extended with several features to enable specific applications to verify, query and filter content in webpages that include:

- Definition of simple rules for website verification and filtering namely, replacing, deleting and blocking content.
- Use of types for static and dynamic verification of rules.
- Consistency checking between rules (one rule cannot violate another rule).

Let's present one simple example which illustrates how a delete rule can be implemented in VeriFLog.

**Example 1.** Given a wiki webpage in an XML document stored in variable *Wiki1*, deleting all the references in the text which do not occur in the bibliography section of that given wiki webpage is done by the following code:

```
delte(ref(3),Wiki1,Wiki2,[not(deep(bibentry((number,R)),_,Wiki1))]).
```

So, if we have the following XML stored in variable *Wiki1*:

```xml
<?xml version="1.0" encoding="utf-8"?>
<WikiArticle>
    ...
    <Content> XCentric <ref>3</ref> is an extension of Prolog with unification of terms of flexible arity which enables a simpler and high level querying and processing of XML data. </Content>
    ...
    <References>
        <bibentry number = "2">SWI-Prolog, http://www.swi-prolog.org/</bibentry>
    </References>
</WikiArticle>
```

By applying the delete rule and since a reference with number 3 is not available in the references at the bottom of the page (attribute number of element bibentry) it will result in a new XML document in variable *Wiki2*:

```xml
<?xml version="1.0" encoding="utf-8"?>
<WikiArticle>
    ...
    <Content> XCentric is an extension of Prolog with unification of terms of flexible arity which enables a simpler and high level querying and processing of XML data. </Content>
    ...
    <References>
```
The replace and failure rules work in an analogous way. The type system allows checking the content against schemas and the consistency checking verifies if one rule is not in violation of another rule, for example, when one rule adds some content which is forbidden by another rule.

3 Visual Editor of Rules

With the tool we describe here a user can select the XML Schema (XSD) [21], describe constraints over documents complying with the given schema and then apply these constraints to instances of that schema. In case the schema is not available, the user can select an XML document and the application will infer the corresponding XSD. It is possible to select sub-trees of the document and apply constraints to its content, such as string manipulation, negation, emptiness and URL checking. It is also possible to introduce constraints manually in order to search elements at arbitrary deep and apply complex constraints to these in a highly declarative and compact syntax. Details and examples are presented next.

3.1 Implementation

This tool is implemented in C# [18] and SWI-Prolog [22]. For the communication between C# and SWI-Prolog we use a third-party library named Swi-cs-pl [19]. In Figure 1 we present the main interface of our application. Here the user can choose between two types of file for the input, an XSD or an XML instance. In the case the user chooses the XML instance, the application infers the related XSD. The user can also choose one of two ways for applying the rules, applying to a unique file or to a directory of files. The idea for choosing a directory is that the user can verify the constraints to a set of files conforming to a given XSD. Note that the XSD was loaded and presented in the left tree view. The user can now proceed by selecting sub-trees and applying rules to these. When applying the constraints, the application first checks if the input file complies with the related XSD.

3.2 Examples

We now present some illustrating examples. For these examples, we use a wiki that stores yellow pages where anyone can contribute and which is available at [23].
**Example 2 (Deleting content).** In these wiki-based yellow pages there is a section where a user can insert a URL with a link to a map showing the location of his/her business, this URL is in a subtree like the one presented next.

```xml
... 
  <map> 
    <label> ... </label> 
    <url> ... </url> 
  </map>
... 
```

The user may, for example, write the content of the `label` element but forget to include the content of the `url` element. We may argue that this doesn’t make sense and impose a rule that checks the content and removes the subtree `map` whenever the `url` element is empty. In Figure 2 we show how this is done. We selected an XML file whose XSD was inferred and presented in the left side. There, the user can select the element to which he wants to apply the constraint. For this example we select the element `map` (the one we want to delete) and click the *Delete* button to open the rule definition window. There, *Element content* dropdown is filled with the elements contained in the subtree of `map`, we can choose any of these and define constraints over their content. These constraints consist in optionally picking the “NOT” checkbox and choosing one of the “Contains”, “Contains valid URL” or “NULL”. In this case we choose the `url` element and pick the “NULL” checkbox. After clicking the *Apply* button the rule is added to the rule...
list and the right tree view is loaded with the new version of the XML file. As shown in Figure 3 the map element does not appear anymore in the final document. The generated rule is presented next:

\[
\text{delete(map(Map), YP1, YP2, [deep(seq(url([],U),empty), Map), (U="empty")])}.\]

Here, \(YP1\) stores the initial XML document and \(YP2\) stores the new XML document after applying the constraint.

**Example 3 (Replacing content).** Given the same wiki webpage presented in the example above, we want to validate if the prices are not missing. If they are missing we want to replace the null content of these with a warning message such as “Prices Unavailable”. Using the visual rule editor, one can select the element \(\text{prices}\) and click the \(\text{Replace}\) button to define this rule, as shown in Figure 4. Here we click the \(\text{NULL}\) checkbox to verify if the content of \(\text{prices}\) is empty. After clicking the \(\text{Apply}\) button the \(\text{Rule List}\) is updated with this rule and the new XML (on the right side) is updated as shown in Figure 5. The generated rule is presented next:

\[
\text{replace(prices([],P), prices([],‘Prices Unavailable’), YP1, YP2, [deep(seq(prices([],P),empty), YP1), (P="empty")])}.\]

Here, \(P\) stores the content of element \(\text{prices}\) and \(YP1\) the initial XML document and \(YP2\) the new XML document after applying the constraint.

Note that rules are being added to the \(\text{Rule List}\) at the bottom of the main window interface. These rules can be all applied to an XML document we choose. Given the following XML file (stored in variable \(YP1\)):
Fig. 3. Result of delete

Fig. 4. Applying a replace
Fig. 5. Result of replace

<body>

<yellowPage>
  ...
  <addressDirections>
    <address>
      129 MacDougall St.
    </address>
    <Map>
      <label>Map it</label>
      <url/>
    </Map>
  </addressDirections>
  ...
  <prices/>
  ...
</yellowPage>

By applying the two rules presented in the examples above the new XML document stored in $YP2$ variable is:

<yellowPage>
  ...
  <addressDirections>
    <Address>
      129 MacDougall St.
    </Address>
  </addressDirections>
  ...
  <prices>Prices Unavailable</prices>
  ...
</yellowPage>
Example 4 (Disapproving webpages). An error found in a document can be seen as so severe that it is better to stop the page processing and present an error message. This could be useful if, for example, this tool was automatically integrated in a website such as Wikipedia to automatically verify errors in content of submitted webpages.

Let’s consider that an invalid email is a severe error. We will implement a simple verification by checking if the email contains an @. If it does not contain an @ we will just present an error message and will not process the XML document. We do this by selecting the Fail button as presented in Figure 6. The generated rule is presented next:

```prolog
fail(YP1,[deep(seq(email([],E),empty),YP1),
          not(sub_string(E,...,')','@'))],'Valid email not found!').
```

Here, YP1 contains the input XML file and the variable E variable contains the email content to verify. In case of error the message “Valid email not found” is shown to the user.

Example 5 (Describing rules manually). Using the basic rules in the interface windows we may be unable to verify every aspect we need. Thus, the editor gives the possibility of manually editing rules in order to use all the power of Prolog and XCentric. Let’s suppose we want to delete all phone numbers which length is not equal to 10. This can be done by clicking the button Edit Manual Rule and inserting the following rule:

```prolog
```
This is shown in Figure 7.

![Manual edited rule](image)

Also, note that it is possible to delete previously defined rules by selecting them and clicking in “Delete Rule” updating the XML in the right tree view with a version where the deleted rule was not applied. It is also possible to save these rules for reusing in the future.

4 Related work

The tool presented here is a visual extension with new features to our previous work presented in [7] and [8]. In [2] the authors presented a rewriting-based framework that uses simulation [12] in order to query terms. In [3], the authors present a semi-automatic methodology for repairing faulty websites by applying a set of concepts from Integrity Constraint [17]. In [10] the author proposed the use of a simple pattern-matching-based language and its translation to Prolog as a framework for website verification. In [20] logic was proposed as the rule language for semantic verification. There the authors provide a mean for introducing rules in a graphical format. In [13] the author proposed an algorithm for website verification similar to [5] in expressiveness. The idea was to extend sequence and non-sequence variable pattern matching with context variables, allowing a more flexible way to process semistructured data. In [16] the authors present
a tool for verification of websites based on a subset of the Xcerpt language [4] and Abductive Logic Programming [15]. A detailed comparison between several approaches to verification can be found in [1].

5 Conclusion and future work

Our tool allows an easy development of rules with constraints to impose over XML content which can be used to automatically verify content in webpages submitted to open collaboration repositories such as Wikipedia. We believe that it can be further extended and used for example, as a browser plugin for constrained content presentation in the client side or to verify quality in terms of design and readability of a webpage.

References

1987.
15. Paolo Mancarella, Giacomo Terreni, Fariba Sadri, Francesca Toni, and Ulle Endriss. The
cifp proof procedure for abductive logic programming with constraints: Theory, implementation and
16. Paolo Mancarella, Giacomo Terreni, and Francesca Toni. Web sites repairing
17. Enric Mayol and Ernest Teniente. A survey of current methods for integrity con-
straint maintenance and view updating. In ER ’99: Proceedings of the Workshops
on Evolution and Change in Data Management, Reverse Engineering in Informa-
tion Systems, and the World Wide Web and Conceptual Modeling, pages 62–73,
aa336809.aspx, 2011.
19. SBS-Softwaresysteme GmbH Uwe Lesta. A CSharp class library to connect .NET
swipica/Generated/Help/introduction.htm, 2011.
20. Frank van Harmelen and Jos van der Meer. Webmaster: Knowledge-based verifica-
tion of web-pages. In Ibrahim F. Imam, Yves Kodratoff, Ayman El-Dessouki, and
Moonis Ali, editors, IEA/AIE, volume 1611 of Lecture Notes in Computer Science,
Comparing application profiles and ontologies for describing experiment data

João Silva
Faculdade de Engenharia da Universidade do Porto, Portugal
joarosilva@gmail.com

Cristina Ribeiro and João Correia Lopes
DEI — Faculdade de Engenharia da Universidade do Porto / INESC Porto, Portugal
{mcr,jlopes}@fe.up.pt

Abstract. Digital data curation is currently becoming an essential part of knowledge management; this holds especially true for scientific data assets, since preserved data can be used for secondary research efforts. Regardless of making their data public or not, some U.Porto researchers show the need for tools which would allow them to deposit their scientific data assets in a secure environment and performing simple analysis, such as temporal series or data sub-setting.

It is in this context that an experiment is being developed at U.Porto, which aims to compare different data models. We compare application profiles against ontologies for the purpose of representing and describing a dataset created from a series of water and sediment pollution control experiments at U.Porto.

We argue that selecting the most appropriate data model and corresponding data exchange format is the first step in offering researchers a system which can provide them with a more consolidated view of their otherwise disperse datasets. As such, the scope of this study is to devise a machine-processable format for the data and its representation information, which are currently only present in human-readable documents produced during the experiment. Such documents include not only description metadata such as geospatial coverage and scientific methodology – variable descriptions, instrument measurement tolerances – but also structural metadata such as the ordering of samples and experiments.

1 Introduction

The ease of access to more powerful means of experimental analysis has made it possible (and sometimes even necessary) to produce more data during the course of scientific research efforts. As a consequence, researchers are producing increasingly large amounts of data, a challenge which falls under the e-Science\footnote{E-Science is computationally intensive science that is carried out in highly distributed network environments, or science that uses immense data sets that require grid computing\cite{1}}

June, 2011- Pages 83–90
domain. However, as the effort put into the production of data increases, so does the need for the adoption of adequate curation practises for this data.

Recent studies have shown the need for the adoption of proper data curation practises and also the need for the implementation of data curation plans. In the USA, for example, NSF grant applicants are required to annex data management plans to their research grant proposals.

This document starts with a description of some challenges in the field of scientific data curation. The production context of the data used as basis for this work is explained, followed by two alternatives developed for its representation. Finally, a comparison of these solutions is presented in the conclusions section.

2 The challenges for Data Curation

Data Curation poses several challenges regarding political, social and ethical concerns. This study focuses on one of the most complex technical issues for data curation solutions: the need to preserve datasets in maintainable, exchangeable formats. Maintainability can be improved through the creation of a public specification of the used exchange format, which must not be dependent on any specific technology. Technologies such as XML, XSD, RDF and OWL are ideal candidates for this purpose because they can not only cope with these needs but also be used to build semantic standards for interoperability which can be reused, either in part or as a whole.

2.1 The pollutant analysis workflow

The Department of Chemistry of the Faculty of Engineering of the University of Porto performs routine analysis and experiments regarding the concentration of certain pollutants in water and sediments, which must be kept under strict limits specified by Portuguese law.

During these analysis runs, samples are taken and analysed using the appropriate apparatuses and experimental methods. This data is then saved in Excel spreadsheets, where it is statistically processed, and the final results are written in Word documents. This workflow is fairly common in many research efforts, and poses obvious preservation concerns. The data is dispersed among several sheets and reports. Careful organisation of these data files must be carried out, often by the researchers themselves. Seeing the problems that arise from the dispersion of their research materials, researchers have expressed the need for solutions which help them centralise and search their data.

---

2 National Science Foundation
3 eXtensible Markup Language
4 XML Schema
5 Resource Description Framework
6 Web Ontology Language
2.2 Alternatives for the representation of research data

In an attempt to mitigate the issues faced by the researchers performing this analysis, we have used two alternatives for the representation of the data produced by these experiments: an application profile, represented as an XML Schema\[5\] document and an ontology, represented in the OWL\[9\] format. This article documents this process and compares the two alternatives.

3 Representing Experiment Data using an Application Profile

The first option for the representation of the gathered data is the development of an XML Schema. XML Schemas can be used to enforce a specific structure for the representation of data. This is important for correct data exchange and essential for the creation of a robust system capable of querying it.

In the development of the schema for this language — named Water Pollution Analysis and Experimental Data Modelling Language (WPAEDML) — some elements from existing schemas were reused, namely: the Dublin Core base elements, Qualified Dublin Core and the CML\[7\] schema.

Dublin Core\[6\] is a widely used format for the representation of Descriptive Metadata — a flat metadata schema for describing a resource. It is used to identify general metadata such as the authors, contributors, or the creation date of a specific resource. A set of 15 elements comprises the Core of the specification.

In addition to the core elements, another Dublin Core specification — DCMI Metadata Terms\[7\] — was used in the proposed schema. This second specification includes qualifiers which can be used, for example, to specify resources related to the one being described, or technical aspects such as the resource’s file format.

To provide a domain-specific element for the analysed chemical substances, the molecule qualifier from the CML specification was used. The CML specification is much more complex and offers sophisticated element such as the description of the atoms and bonds that make up the structure of a molecule\[14\]. In this case, however, such level of detail is not required, since only the molecule name is required to uniquely identify a given substance. The advantage of using part of an existing specification whenever possible is that by sharing the same qualifiers, better interoperability between data models can be attained. The final result is shown in Figure 1.

3.1 The developed schema

In this schema, a run is a set of experiments, which can be seen as a table with its associated metadata. The run element is at the root of this hierarchy along with its own descriptive metadata. This metadata is represented by a

\[7\] Chemical Markup Language\[13\]
convenience group specified in the DC Core schema, which provides a link to its set of 15 base elements.

Each experiment has a series of results, which can be seen as the rows of a table; the header names for this conceptual table are the attributes of the result element. The analysed_molecule attribute’s datatype, moleculeIDType, is specified in the CML schema to identify molecules by their name. The unit attribute also comes from the CML schema and is used to specify the unit of measurement for the result.

The coverage element is taken from the Dublin Core specification and means to specify the geospatial position where the data for this experiment was gathered.

The related_things node contains a list of related resources referenced by the experiment — for example, the Portuguese laws that specify the legal limits for any analysed substances — and any other resources which reference the annotated resource.

The tested_molecules element contains the list of substances analysed throughout the whole run and is taken from the CML Schema. The methodology element contains an identifier to another resource containing the information about the scientific procedures and methods followed during the run.

Finally, the formats node lists all the file formats in which the experiment results are available.

3.2 Metadata Granularity

As the schema implies, there are different levels of metadata at different granularities inside the dataset. There is metadata at the run level, represented by the formats, methodologies, tested_molecules and related_things elements. At the experiment level there are several attributes which are simple datatypes and domain-dependant — listing them would not add to this study. Lastly, the columns names — symbolised by the attributes — which are specified at the result level can be considered metadata themselves.

4 Representation of Experiment Data through an Ontology

The specification of an XML schema can be used to specify a syntax for the exchange of this type of data. However, ontologies can add richer semantic content to the data representation. In this case, concepts like properties can help establish the semantics of all relationships between different parts of datasets — something which is not present in an XML Schema.

Following the principles of the Semantic Web[8], we have reused Classes and Properties present in three public ontologies, linking these concepts with those which are specific to the representation of this dataset.

The reused ontologies are the Dublin Core RDFS specification, the Measurement Units Ontology and the ChemInf Ontology, provided by semantic chemistry[10], an open project to support semantic chemistry.
Fig. 1. The structure of the developed XML schema
Figure 2 offers a visual representation of the developed ontology.

In this ontology, the Experiment, Run and Result classes are specific to this study. All others are reused classes from existing ontologies.

A Run represents the group of Experiments. It is materialised in a data file which has a format represented by the MediaTypeOrExtension class from the Dublin Core Terms ontology, encapsulating a MIME type. It also includes analysis on a set of chemical substances. These substances are represented by the CHEMINF_000066 class, which is specified in the Cheminf ontology as the representation of “an information entity which is about a polyatomic molecular entity”[11]. Runs must state the method through which all their experiments are produced, providing a placeholder for researchers to annotate their research, including relevant parameters such as equipment measurement tolerances or analysis procedures. This method is represented through the Provenance Statement class from the Dublin Core ontology.

An Experiment is the representation of the analysis’ results on a single sample of water or sediment. It is part of a Run, and yields a series of results. To perform the appropriate matching between the experiment and the place in which the samples were collected, the Location class from the Dublin Core Terms ontology.

Finally, a Result is always produced in the context of an Experiment. It represents the measured concentration of a substance, represented by the CHEMINF_000066 class and expressed in an UnitOfMeasurement, taken from the Measurement Unit Ontology[12].
4.1 Similarities between the XML Schema and the Ontology

XML Schemas and ontologies cannot be seen as comparable entities since they reside in different levels of abstraction. However, there is an implicit correspondence between some Schema elements and some ontology classes. The most relevant example is the result - experiment — run element hierarchy present in the XML schema, which is similar to the Result — producedIn — Experiment and Experiment — isPartOf — Run properties in the ontology. This kind of relationship has been analysed in studies looking to extract semantic information from XML Schemas[17].

5 Conclusions and Future Work

An XML Schema such as the one presented in this study can help represent the analysed experimental data in a machine-processable format — something Excel spreadsheets are not suited for. This can be is a critical first step in the creation of a system capable of proper data curation.

The creation of an ontology for this data can open the data to the world, since Classes and Qualifiers can be taken from existing ontologies, allowing for the use of shared semantics in the representation of the experimental data.

The presented approaches are useful in the context of Data Curation since they can help solve two of its main issues: the need for shared, domain-specific data models to correctly represent data and the need for agreement on the semantics of data representation.

Research was performed during the course of this work to find existing ontologies and schemas which could be reused. From this research, we have concluded that schemas are more easily found than ontologies. The semanticweb.org website[15], for example, does not yet list the ontologies used for this work.

Former research[16] has shown that in many cases, it is possible to establish relationships between ontologies and XML Schemas. These matches are also present in this study, since there are similarities between parts of the XML Schema hierarchy and a set of Properties and Classes in the developed ontology.

We conclude that these two solutions complement each other in a data curation environment. An XML Schema is useful to specify a data exchange format for the raw data, which is a part of deposit policies for a data curation solution. It is also easier to specify and use than an ontology, but lacks its semantics.

Future work on this subject includes the analysis of potential applications of the two presented approaches in an hypothetical production environment — for example, a scientific data repository.

References

2. Phillip Lord, Alison Macdonald et al.: From Data Deluge to Data Curation, 2004
   http://www.ukoln.ac.uk/ukoln/staff/e.j.lyon/150.pdf - Consulted on April 2011
9. W3C: OWL 2 Web Ontology Language - Document Overview http://www.w3.org/TR/owl2-overview/ - Consulted on April 2011
An Engine for Generating XSLT from Examples

José Paulo Leal¹ and Ricardo Queirós²,

¹CRACS & DCC-FCUP, University of Porto, Portugal
zp@dcc.fc.up.pt
²CRACS & DI-ESEIG/IPP, Porto, Portugal
ricardo.queiros@eu.ipp.pt

Abstract. XSLT is a powerful and widely used language for transforming XML documents. However its power and complexity can be overwhelming for novice or infrequent users, many of which simply give up on using this language. On the other hand, many XSLT programs of practical use are simple enough to be automatically inferred from examples of source and target documents. An inferred XSLT program is seldom adequate for production usage but can be used as a skeleton of the final program, or at least as scaffolding in the process of coding it. It should be noted that the authors do not claim that XSLT programs, in general, can be inferred from examples. The aim of Vishnu - the XSLT generator engine described in this paper – is to produce XSLT programs for processing documents similar to the given examples and with enough readability to be easily understood by a programmer not familiar with the language. The architecture of Vishnu is composed by a graphical editor and a programming engine. In this paper we focus on the editor as a GWT web application where the programmer loads and edits document examples and pairs their content using graphical primitives. The programming engine receives the data collected by the editor and produces an XSLT program.

Keywords: XSLT, Transformations, Refactoring.

1 Introduction

Generating a XSLT program from a pair of source and target XML documents is straightforward. A transformation with a single template containing the target document solves this requirement, but is valid only for the actual example. Using the information from the source document we can abstract this transformation. The simplest way is to assume that common strings in both documents correspond to values that must be copied between them. If we explicitly identify these correspondences we can have more control over which strings are copied and to which positions. However, a transformation created in this fashion is still too specific to the examples and cannot process a similar source document with a slightly different structure. For instance, if the source document type accepts a repeated element  and the example has  repetitions of the element then the generated program would accept exactly  repetitions of that element.
Although too specific, a simple XSLT program can be used as the starting point for generating a sequence of programs that are more general and are better structured, ending in a program with a quality similar to one coded by a human programmer. To refine an XSLT program we can use second order XSLT transformations, i.e. XSLT transformations having XSLT transformations both as source and target documents. In this approach the role of an XSLT generation engine is to receive source and target examples, and an optional mapping between the strings of the two documents, generate an initial program and control the refinement process towards the final XSLT program.

The aim of this paper is the presentation of Vishnu – an XSLT engine for generating readable XSLT programs from examples of source and target documents. Readability is an essential feature of the generated programs so that they can be easily understood by a programmer not familiar with the language. The architecture of Vishnu is composed by a graphical editor and a programming engine. The former acts as a client where the programmer loads and edits document examples and pair their content using graphical primitives. The latter receives the data collected by the editor and produces an XSLT program.

There are several use cases for an XSLT generation engine with these features. The Vishnu generator was designed to interact with a component that provides text editing functions for the end-user or programmer. A client of Vishnu can be a plug-in of an Integrated Development Environment (IDE) such as Eclipse or NetBeans. In this case the IDE provides several XML tools (highlighting, validation, XSLT execution) and the plug-in is responsible for binding the content of text buffers and editing positions with the engine and retrieving the generated XSLT program. Vishnu can also be used as the back-end of a web environment for XSLT programming. In this case the web front-end is responsible for editing operations and invokes engine functions for setting the example documents and mappings, and retrieving the generated program. The generator can also be used as a command line tool as part of a pipeline for generating and consuming XSLT programs. In this last case the generator processes example documents in the local file systems, making mostly use of default mappings.

The rest of the paper is organized as follows. Section 2 presents work related to XSLT editing and generation. In the following section we present the inner structure of the XSLT generator that is composed of three main components: the context, the generator and the refiner. Then, we evaluate the Vishnu XSLT generation engine from three complementary and interrelated approaches, focusing: the consistency of generation and refinement process; the coverage of the existing rules; and the adequacy of the Vishnu API to XSLT editing environments. Finally, we conclude with a summary of the main contributions of this work and a perspective of future research.

2 Related Work

The first step to start editing XSLT files is choosing the editor that most suits one’s programming environment. There are tools integrated in XML IDEs [1, 2], tools
integrated in general purpose IDEs as plug-ins [3, 4, 5, 6, 7, 8] and even standalone applications [9, 10, 11]. Despite the existence of several environments for programming in XSLT, usually integrated into IDEs, they do not use visual editing for programming. Moreover, as far as we know, none of the graphical XSLT programming environment generates programs from examples as source and target documents.

Hori and Ono [12, 13] use an example-based annotation tool which relies on a target document editor. The main concepts of their approach are depicted in Figure 1. An annotator can edit a target document (e.g., an HTML page) by using the capabilities of a WYSIWYG authoring tool (1). The editing actions are recorded into an operation history (2). When the editing is finished, the annotation generator creates transformational annotation for the document customization (3), which can be further used by XSLT processor to replicate the transformation from the initial document to the customized document.

![Fig. 1 History based document transformation.](image)

Spinks [14] presents an annotation-based page-clipping engine providing a way of performing Web resources adaptation. At content delivery time, the page-clipping engine modifies the original document based on: 1) the page-clipping annotations previously generated in a WYSIWYG authoring tool and 2) the user-agent HTTP header of the client device. The page-clipping annotation language uses the keep and remove elements in the annotation descriptions to indicate whether the content being processed should be preserved or removed.

### 3 The Vishnu engine

The Vishnu engine [15] concentrates all the tasks related with the automatic generation of an XSLT program from examples using second order transformations. Nevertheless, it was designed to interact with a client. A client of the Vishnu engine concentrates all the tasks related with user interaction where the programmer loads and edits document examples and pairs their content using graphical primitives.

The communication between these two components is regulated by the Vishnu API. Hence, the architecture of the Vishnu application is composed by a Graphical Editor and a Programming Engine as depicted in Figure 2.
The former acts as a client where the programmer loads and edits document examples and pairs their content using graphical primitives. The design and implementation of a client for the Vishnu engine is presented in the next section to validate the adequacy of the Vishnu API to XSLT editing environments.

The latter receives the data collected by the editor and produces an XSLT program. The engine relies on the Vishnu API that includes methods for setting the source and target documents as streams of characters, setting a mapping between the strings of these documents using editing locations (offsets), and retrieving the resulting XSLT program. The Vishnu API includes also functions for supporting graphical interaction in the editor and for configuring the generation process. The functions for selecting strings in the XML documents (text and attribute nodes) from editing locations are example functions for supporting graphical interaction. The Vishnu façade class implements this API and hides the inner structure of the XSLT generator that is composed of three main components: the **context**, the **generator** and the **refiner**.

### 3.1 Context

The central piece of the engine is the generation **context**. The context holds the source and target documents and the mapping between the two and is responsible for converting between the external textual representation provided by the client and the internal XML representation required by the Vishnu. In particular this component is responsible for converting document position into XPath expressions and vice-versa.

The conversion is managed by the PathLocator class. This class converts text locations (offsets) into **IdPaths** expressions and vice-versa. An **IdPath** is an absolute XPath expression which selects either single texts or attribute nodes in an XML document. The general form of an **IdPath** is:

\[
/n^1[p^1]/\ldots/n^n[p^n]/\text{text()}
\]

\[
/n^1[p^1]/\ldots/n^n[p^n]/@\text{attr}
\]

It should be noted that locating nodes from using their editing positions and the reverse are not operations supported by the APIs for processing XML documents.

The Context component is also responsible for the generation of the mapping between the source and the target documents. It maintains an XML map file.
identifying the correspondences between both. These identifications can be inferred automatically or manually set through the Editor. The following XML excerpt shows an example of a source, target and a list of pairs of XPath expressions relating them merged in a file called vishnu.xml.

```xml
<vishnu xmlns="http://www.dcc.fc.up.pt/vishnu">
  <!--Source document -->
  <source>
    <rss version="2.0" xmlns="http://backend.userland.com/rss2">
      <channel>
        <title>News</title>
        <link>...</link>
        <description>...</description>
        <item>...</item>
      </channel>
    </rss>
  </source>
  <!--Target document -->
  <target>
    <html xmlns="http://www.w3.org/1999/xhtml">
      <head>
        <title>News</title>
      </head>
      <body>
        <h1>News</h1>
        ...<h1>News</h1>
      </body>
    </html>
  </target>
  <!--Pairing document--> 
  <pairings>
    <pairing
      source = "//rss[1]/channel[1]/title[1]/text()"
      target = "//html[1]/head[1]/title[1]/text()"/>
    <pairing
      source = "//rss[1]/channel[1]/title[1]/text()"
      target = "//html[1]/body[1]/h1[1]/text()"/>
  </pairings>
</vishnu>
```

This file will serve as input for the Generator component to produce a XSLT program.

### 3.2 Generation

The purpose of the **generator** is to produce an initial XSLT program from the source and target, using a string mapping. If no mapping is provided by the client then it uses a default mapping inferred by the context component, linking text or attribute nodes in both documents with equal character strings. The generator component
receives as input the paring file and, using a second order transformation, produces a specific XSLT program. As an illustration we present the output of this second order stylesheet based on the example included in the previous subsection.

```xml
<xsl:template match="/">
  <html>
    <head>
      <title>
        <xsl:value-of select="/vishnu/source/rss[1]/channel[1]/title[1]/text()"/>
      </title>
    </head>
    <body>
      <h1>
        <xsl:value-of select="/vishnu/source/rss[1]/channel[1]/title[1]/text()"/>
      </h1>
      ...
    </body>
  </html>
</xsl:template>
```

The initial XSLT program has a single template containing an abstraction of the target document. To abstract the target document the target positions in the mapping are replaced with `xsl:value-of` instructions referring corresponding source positions in the mapping. As explained previously, with this level of abstraction the initial transformation is only able to process a document with the exact same structure of the source document provided as input. To be of any practical use this program is submitted to a refinement process.

### 3.3 Refinement

The refinement process produces a sequence of XSLT programs $\rho_n$ starting with the initial program $\rho_0$ by applying $R = \{r_i\}$ set of second order XSLT transformations called refinements. Refinements can be divided in two categories: simplifications and generalizations.

Let $S_0$ and $T_0$ be respectively the example source and target documents. All refinements $r_i$ have the following invariant: $\rho_n(S_0) = T_0 \Rightarrow r_i(\rho_n)(S_0) = T_0$ that is, if a program maps the example source document to the example target document then the refined program has the same property. A simplification refinement is even more restrictive and any document $S$ that is converted by program $S_0$ is equally converted by its refinement, i.e. $\forall S, T \rho_n(S) = T \Rightarrow r_i(\rho_n)(S) = T$. Simplifications are “safe” refinements but fail to introduce the level of abstraction needed for a transformation to be effective, hence this stronger requirement is relaxed for abstractions.

An example of a generalization is the refinement that unfolds a single template into a collection of smaller templates. Candidates to top elements in the new template are elements whose XPath expressions in `xsl:value-of` share a common and non-trivial prefix that can be used match of the new template. As it introduces new
templates with relative expressions in the match attribute this refinement is not a simplification. The new template may match with nodes with the same tag occurring in different points in a different source document structure. To minimize the chance of unwanted matches this refinement associates a mode to the new template that is used also by the xsl:apply-template instruction that invokes it. An example of a simplification is the refinement that removes redundant modes from xsl:template and xsl:apply-template instructions. This refinement selects templates with non empty modes that cannot be matched by other templates. That mode is removed both from the selected template and all xsl:apply-template referring it. The current Vishnu implementation includes over 10 refinements.

As an illustration we present the final output of the refinement process based on the example included in the previous subsection.

```xml
<xsl:stylesheet version="1.0"/>

<xsl:template match="rss2:channel">
  <xhtml:html>
    <xsl:apply-templates mode="xhtml:head" select="rss2:title"/>
    <xhtml:body>
      <xsl:apply-templates mode="xhtml:h1" select="rss2:title"/>
      <xhtml:ol>
        <xsl:apply-templates select="rss2:item"/>
      </xhtml:ol>
    </xhtml:body>
  </xhtml:html>
</xsl:template>

<xsl:template match="rss2:item">
  <xhtml:li>
    <xhtml:a href="{rss2:link}">
      <xsl:value-of select="rss2:title"/>
    </xhtml:a> -
    <xsl:apply-templates select="rss2:description"/>
  </xhtml:li>
</xsl:template>

<xsl:template match="rss2:description">
  <xhtml:i><xsl:value-of select="."/></xhtml:i>
</xsl:template>

<xsl:template match="rss2:title" mode="xhtml:h1">
  <xhtml:h1><xsl:value-of select="."/></xhtml:h1>
</xsl:template>

<xsl:template match="rss2:title" mode="xhtml:head">
  <xhtml:head><xsl:value-of select="."/></xhtml:head>
</xsl:template>

</xsl:stylesheet>
```
The Vishnu engine supports different refinement strategies to control the application of the refinement set \( R \). A refinement strategy indicates the next refinement to use is informed if the suggested refinement has changed the XSLT program and decides when the refinement process is complete. There are several refinement strategies that can be set using the Vishnu API. The most effective strategies implemented so far apply the refinements in a predefined order, repeating the application of refinement while it is effective.

4 Validation

The Vishnu engine was validated in three complementary and interrelated approaches, focusing on:

- **consistency** of the generation and refinement process;
- **coverage** of the existing rules;
- **adequacy** of the Vishnu API to XSLT editing environments.

By default Vishnu validates the **consistency** of the generation and refinement process by checking that each intermediate transformation converts the example source document into the examples target document. If this invariant is not satisfied then the refinement process is aborted and an error is reported to the client.

To validate the **coverage** of the existing rules different scenarios were created. Each scenario includes source and target document and a mapping, as well as the expected program.

![Diagram](image)  
**Fig. 3** The RSS to HTML scenario.
The manipulation of a scenario in Vishnu is made by the Scenario class. This class provides a set of methods for testing the Vishnu engine. Typical uses involve a set of scenarios where for each scenario the generated output of the engine is matched with the resources enclosed on the scenario itself. The current scenarios include the conversion of: 1) RSS documents to HTML; 2) Mathematical expressions in MathML to presentation MathML and 3) Meta-data in LOM (Learning Object Metadata) to RDF. The Figure 3 shows the inner workflow used for testing the RSS to HTML scenario. A mixed-content scenario has not been added yet since the context component is not supporting indexes in text nodes.

To validate the adequacy of the Vishnu API we developed a simple web environment for XSLT programming based on the Google Web Toolkit (GWT), an open source framework for the rapid development of AJAX applications in Java. When the application is deployed, the GWT cross-compiler translates Java classes of the GUI to JavaScript files and guarantees cross-browser portability. The specialized controls are provided by SmartGWT, a GWT API's for SmartClient, a Rich Internet Application (RIA) system.

The graphical interface of the front-end is composed by two panels: Mapping and Program. In the Mapping panel the "programmer" uses graphical tools to map strings in two XML documents corresponding to a source and a target documents for the intended XSLT transformation. In the Program panel the user obtains the resulting XSLT and can continue editing it.

Figure 4 shows the RSS-to-HTML scenario being used on the Vishnu client GUI with its main components labelled with numerals. The Mapping panel includes two side-by-side windows for editing respectively (1) the source and (2) the target documents. These documents may be created either from scratch or based in scenarios predefined in the Engine. Regardless of the choice the correspondences between both can be set (3) manually through the Editor or inferred by the Engine.

When setting correspondences manually the programmer is able to pair contents on these windows by selecting and highlighting with color texts where the origin is on the source document and the destination is on the target window. Origin and destination must be character data, either text nodes or attribute values.

When automatic correspondence is used Vishnu identifies pairs based on: text matches (text or attribute nodes) or text aggregation. In the first mode strings occurring on text and attribute type nodes on the source document are searched on the text and attribute nodes of the target document, and only exact matches are considered. In the second mode Vishnu aggregates strings in the source document to create a string in the target document. After automatic pairing, the inferred correspondences are presented in the GUI with colors mapping the two XML documents. The user can then manually reconstruct the pairing of string between both documents.
In complement to creating the source and target documents from scratch, the user can fill in automatically the two rich text editors by using scenarios (4). Each scenario includes source and target document and a mapping, as well as the expected program.

5 Conclusions

In this paper we present Vishnu - an XSLT generator engine that aims to produce XSLT programs for processing documents similar to the given examples and with enough readability to be easily understood by a programmer not familiar with the language.

The project that lead to the development of the Vishnu can follow different paths: the engine can be used in other XSLT programming environments; the API of the engine can extended with new functions; and the refinement process can be extended with new refinements. First of all, the Vishnu API was validated with a web environment but the appropriate place to apply it would be an IDE with support for XML. Eclipse is particularly suited for this purpose because it is not a XML IDE but rather an IDE for programming in general with tools for handling XML, including XSLT programming. Secondly, the Vishnu engine was designed as a tool for generating simple XSLT programs from examples and can be extended for other uses. The refinement process was designed to improve the quality of a naïve XSLT program automatically generated from examples but can be used to improve any XSLT program. In fact, an interesting side effect of this research is the definition of...
sort of “canonical XSLT” in terms of second order XSLT transformations. In practical terms we plan to expand the Vishnu API to enable the use of the refinement process on a given XSLT program, rather than only on those generated from examples. This feature may be used in the XSLT programming environment to refactor any XSLT programs, including the generated program after it was edited by the programmer. Finally, Vishnu is an expandable system in the sense that refinements and refinement strategies can be easily integrated. We expect to create new refinements both to improve the quality of automatically generated XSLT programs and to introduce new forms of automatically refactoring existing XSLT programs.

References

xml2pm: A Tool for Automatic Creation of Object Definitions Based on XML Instances

Nuno Carvalho¹, Alberto Simões², and José João Almeida¹

¹ Departamento de Informática, Universidade do Minho
{narcarvalho,jj}@di.uminho.pt
² Centro de Estudos Humanísticos, Universidade do Minho
ambas@ilch.uminho.pt

Abstract. The eXtensible Mark-up Language (XML) is probably one of the most popular markup languages available today. It is very typical to find all kind of services or programs representing data in this format. This situation is even more common in web development environments or Service Oriented Architectures (SOA), where data flows from one service to another, being consumed and produced by an heterogeneous set of applications, which sole requirement is to understand XML.

This workflow of data represented in XML implies some tasks that applications have to perform if they are required to consume or produce information: the task of parsing an XML document, giving specific semantics to the information parsed, and the task of producing an XML document.

Our main goal is to create object definitions that can analyze an XML document and automatically create an object definition that can be used abstractly by the application. These objects are able to parse the XML document and gather all the data required to mimic all the information present in the document.

This paper introduces xml2pm, a simple tool that can inspect the structure of an XML document and create an object definition (a Perl module) that stores the same information present in the orinial document, but as a runtime object. We also introduce a simple case of how this approach allows the creation of applications based on Web Services in an elegant and simple way.

1 Introduction

In todays’ distributed world of cloud computing and a multitude of approaches for sharing and distributing resources [1, 8], the need for exchanging information between heterogeneous independent systems has become a necessary evil [11, 7].

This interoperability between systems requires information interchange, and to make this information sharing possible and reliable a lot of methods and techniques have been already proven worthy: from the basic RPC (remote procedure call), CORBA (Common Object Request Broker Architecture), or Java RMI (Remote Method Invocation), to the most recent web-oriented approaches, like SOAP web-services (Simple Object Access Protocol) or REST-less (Representational State Transfer) services [5, 12, 13, 6].

Independently of which approach is adopted, a common challenge always ends up being addressed: how to share the information, in a persistent and understandable way.
A sane approach for this problem is to use a structured and well defined document where some basic semantic information about the data can also be included. XML [2] is a mark-up language that has been proven as a good choice to achieve good results on data interchange. This technology is widely available in most development environments (a wide range of tools like parsers, checkers, pretty-printers, and others, are already available), and it exists for quite some time now, which means that it is mature.

Although XML can be an excellent data carrier between heterogenous systems, once the information reaches the application level, processing and handling it in its raw format can be painful and have a deep impact on programming performance (and sometimes, execution performance). Typically it is necessary to transform the data representation maintaining the information content and its semantics.

This transformation commonly requires a task in the parsing family of operations, which will probably be common for every time an application requires to use information stored in XML format. Figure 1 tries to illustrate these common tasks that are performed by applications before they start solving whatever problem they are trying to solve.

Notice that this data structure transformation is not required just because XML is a textual format and during runtime programmers prefer to have dynamic structures. Most of the times, the resulting structure will not be used just to make the data usable by the application. It will be the place where data semantics will be analyzed (validating data types, for instance), and where data will evolve through time. Figure 1 also illustrates an Arbitrary Processing stage, this is because most of the times another set of tasks needs to be performed after the parsing stage: for example making sure information is standardized, text fields are written in a specific character set, time stamps are all in the same format, etc.

After processing the data, applications want to serialize it again, so it can be sent to other services (or to the service requester). That is, the application needs to convert its data back to XML format, involving yet another usual task on applications that deal with XML formats.
This pattern can be found in many applications, disguised in one implementation or another, but is present most of the times. Since this transformation action is such a common issue why not envisage a systematic and automatic solution to deal with it?

In this article we propose the use of objects, in the context of Object Oriented Programming (OOP) [14], to store the data exchange via XML in runtime, and introduce a tool that can automatically create the required code to create objects that mirror the information and behavior of a XML document. These objects main responsibilities are: to parse the XML data and populate objects proprieties with information; and provide accessor and setters that allow applications to read and update data.

2 Related Work

There are tools already available that allow a similar approach to the one described in this article. A few of these tools are enumerated here and are very briefly compared to the xml2pm. We are not interested in tools that act at runtime, we are only interested in tools that are using during development time to create the required object definitions and parsers. This is mainly because performance issues and because we want the user to be able to add arbitrary tasks to the methods in the object definitions. Some of these tasks can even involve chaging information semantics.

- autoXML[10] generates a parser for an XML document given a DTD file. Besides a parser the required structures to mirror the document in memory are also created. This is very similar to the tool introduced in this paper, except for two minor differences: the structures created are not objects per se in runtime, and the user is required to manually call the parser.
- JAXB[9] is another tool that is able to create classes definitions from schemas that can be instantiated from XML documents.

A more comprehensive and complete list is being maintained by Ronald Bourret[3]. Most of the tools available are either for Java or for C, to integrate this work in another project dealing with ontologies a implementation of these objects in Perl was required, that was one of the major initial motivations for this work.

2.1 Design Goals

This section describes the complete set of design goals that motivated the work described in this paper. Most of the tools referenced in the previous section and their analysis were used as a starting point for devising these, but we feel that none of them by itself could fulfill the entire set.

- Objects definitions are created during application development.
- Objects are able to parse XML documents and mimic the information by themselves, no additional stages are required to be manually done.
- Objects are able to produce an XML document representing their current data.

3 http://www.rpbourret.com/xml/XMLDataBinding.htm
- Easy to add arbitrary extra processing tasks to information retrieval and update methods.
- Objects can change information semantics more suitable to applications needs.
- Independent objects and parsing capabilities, so that nested structures in a XML document can give origin to their own objects that can be used independently.
- Bigger concern with quickly processing smaller documents than being able to handle huge amounts of data.
- Do not require schemas for the XML documents, this is mainly because most of the times schemas are not available.
- And finally, objects available in the Perl programming language so this tool can be used integrated with the CROSS web portal architecture.

3 Prototype Tool

As described earlier the main goal of this tool is to allow the use of objects by applications to reach and manipulate information shared in XML. This means that the new prelude for operations would be something similar to what is illustrated in figure 2.

In order to do an experimental validation of this approach we developed a prototype tool that can be used to create Perl modules (Perl has no native support for the traditional objects in the context of object oriented programming language, so modules are used to implement them). The tool is named xml2pm, in the sense that it processes an XML file and produces one or more Perl modules with the required code to manipulate XML documents with that specific structure.

To use this tool we simply execute it giving the name of an XML file as argument, as shown on figure 3.

---

*Fig. 2. Application workflow alternative prelude.*

4 http://twiki.dii.uminho.pt/twiki/bin/view/Research/CROSS/WebHome
In this particular example we processed an XML file that stores the information for a Short Message Service (SMS) message. A file called Sms.pm (based on the XML filename sms.xml) is created (pm is the typical extension for a Perl module file).

This module includes the code to create and use an object, that has the required attributes and methods to provide the information present in any XML document with that same format. This includes all the code required to parse a XML document and populate the object, as well as the code needed to serialize that information again in XML.

To illustrate the usage of this tool with a very simple example, consider the XML document shown in figure 4. There are four attributes that are required to describe a message: the sender, the receiver, the date and the body of the message itself. Therefore this is the minimum set of attributes that new objects needs to have. After processing the XML file a Perl module will be generated. Its main structure is shown in figure 5.

Looking up to the code, we can see that the new object type is called Sms, it has a new method for creating new instances of this object, and a set of setters/accessors.

The constructor (the new method) is used for parsing the source XML and setting the objects’ attributes with the corresponding values. The tool is able to fetch a file from the filesystem,

```perl
$sms = Sms->new(file => 'sms.xml');
```

or from an URL (Universal Resource Locator):

```perl
$sms = Sms->new(url => 'http://randomsms.org/fetch/random');
```
Fig. 5. Perl code to implement the interface to SMS file format.

Setters and getters are created for each attribute. These methods can be called without any arguments, acting as a getter, returning the actual value for that attribute. When an argument is passed, they act as setters, changing that attribute content.

Finally, a serializing function is also created. Its name is to_xml and it returns the object data in XML format. It can also accept some extra parameters, so the XML is written directly to a file.

This object has the same attributes than the original XML document but from the programmer’s point of view it is much easier to manipulate in runtime than the XML textual version. The automatic creation of these objects definition is one of the major advantages of using this kind of tools.

Instead of developing a library that, in runtime, analyses the XML file and creates a generic object from it, we preferred to generate static code, that can be used anytime, in the same application or in any other that needs to manage the same kind of data.

This approach has two major advantages: in runtime the information is stored in an object (or a set of objects), instead of an XML document, and the object creation is performed automatically and only once for a specific XML instance. The developer can also easily change the object definition to add extra processing tasks for specific attributes or even change the semantics of the information if required.

Getting back to our previous text message example, we can add an extra action that always make sure that the name value stored in the runtime object is capitalized, as illustrated in figure 6. Now, every time this module is used, this behavior will stick. This ensures flexibility, as we are no longer looking just to a object serialization tool.

4 Case Studies

This section will present two bigger examples:

- The first one is merely a more complex example than the illustrative example from the previous section, where the XML file includes a more complex structure, and there are repeating elements (lists);
- The second one, is a more realistic example, that uses xml2pm to quickly create an interface to an XML web service [4].
sub sender {
    my $sender = shift;
    $self->{sender} = shift if @_;

    # added transformation
    $self->{sender} = "s/\W+/ucfirst($1)/ge;

    return $self->{sender};
}

Fig. 6. Modified setter/getter to ensure name capitalization.

4.1 Processing Music Catalogs

In this first example we will illustrate the use of the xm2pm tool with a little more complex XML structure than the one shown in section 3. In this example, XML documents are used to represent catalogs of music albums. Each catalog has a name and an associated creation date. The collection consists of a list of albums. For each album we have its title, the name of the artist, the company that edited it and in which year it was released. Figure 7 illustrates an example XML document that represents a music catalog.

```xml
<?xml version="1.0"?>
<catalog>
    <name>The 80s Collection</name>
    <created>1980-12-31</created>
    <collection>
        <item>
            <title>Empire Burlesque</title>
            <artist gender="male">Bob Dylan</artist>
            <company>Columbia</company>
            <year>1985</year>
        </item>
        <item>
            <title>Like A Virgin</title>
            <artist gender="female">Madonna</artist>
            <company country="USA">Warner Bros</company>
            <year>1984</year>
        </item>
    </collection>
</catalog>
```

Fig. 7. catalog.xml file contents.

The first step is to call our xm12pm application, feeding in the XML sample document, as illustrated in figure 8. Note that in this case the tool did not create just a
module, but a pair of modules. This is xml2pm behavior when the XML document has lists.

```perl
$ xml2pm catalog.xml
Processing catalog.xml ..ok!
Writing Catalog.pm .. ok!
Writing Item.pm .. ok!
```

**Fig. 8.** Creating a named module from a Catalog XML document sample.

The tool created two modules. Catalog.pm implements the main XML object. But when it gets to its `collection` element, a list of items is present. Each one of those items can be seen as small XML document (which root element is `item`), and that is why the tool creates a `Item.pm` module as well. It will handle the data for each item, and Catalog.pm element `collection` will deal with lists of this kind of objects. Figure 9 illustrates the Catalog.pm file.

```perl
package Catalog {
  use Item;

  sub new {
    ...

    if ($name eq 'item') {
      my $item = Item->new($field);

      push @{$self->{catalog}}, $item;
    }
    ...
  }

  sub name    { ... }
  sub created { ... }
  sub collection { ... }
  sub to_xml  { ... }
}
```

**Fig. 9.** Perl code to implement the interface to Catalog.

A closer look at the `new` function, which is responsible for creating a new object that represents a catalog, shows the need to handle the collection, nested list of items.

Figure 10 shows the code to handle `item` elements. This example includes an example of how attributes are being handled at the moment (see the `artist` method).

The `new` function is responsible for populating the initial instance and the object, and attributes are later available in a finite function key name to value, which is also stored, besides the value of the element.
package Item {
    sub new { ... }
    sub title { ... }
    sub artist {
        my $self = shift;
        my ($value,$attributes) = @_; if @_;:
        $self->{artist}->{value} = $value;
        $self->{artist}->{attributes} = $attributes;
        return $self->{artist};
    }
    sub company { ... }
    sub year { ... }
    sub to_xml { ... }
}

Fig. 10. Perl code to implement the interface to Item in a collection.

In this specific case, the artist method works differently than the previous examples. The setter do not receive just the value to be set, but also a reference to an associative array (hash reference), with the element’s attributes. When used as an accessor, the return value is, also, an associative array, with the key value being used for the element’s content, and the key attributes to the associative array of attributes.

We are not very happy with this approach for handling arguments, because sometimes it can result in some confusing code in the applications using the module. We are currently working on other approaches to handle attributes without changing the default accessor/setters.

Using these modules we can now create simple programs that process information from these kind of catalogs, and produce different results. As a simple example, consider the code in figure 11 an application example to present the catalog in HTML format. This code in elegant and simple, easy to read and maintain.

Now that we have a better idea of what this tool can do, the next section introduces a more practical application example, to clearly show how the use of xml2pm can save time, and increase the elegance and maintainability of implemented applications.

### 4.2 Quick Generation of Web Services Clients

A Web Service is a very common case where the XML format is used to transport data. In this section we will show in a set of simple steps how to take advantage of the xml2pm tool to implement an application that relies on the information provided by a web service.

We will use an Web Service that, given a city (and respective country) returns, among other information, its geographical position (latitude and longitude).
To bootstrap, we need a sample XML file as returned by the web service. There are different ways to query the web service and store the resulting XML file. One option would be the use of the `curl` command available on most Linux/Unix operating systems. This can be accomplished as follows:

```
curl http://.../GeoLookupXML/index.xml?query=braga > location.xml
```

Having a sample of the XML that the web service provides we can build our code using `xml2pm`. We can also use a switch on the command to give it a proper name, as shown in figure 12.

```
$ xml2pm -n Geo::Location location.xml
Processing location.xml .. ok!
Writing Geo::Location .. ok!
```

**Fig. 12.** Creating a named module from a sample web-service response file.

A new module named `Geo::Location`, that implements objects with that same name, can now be used to query that specific web service in a clean object-oriented fashion. Figure 13 shows a simple application that prints latitude and longitude for a city which name was passed as parameter in the command line (in this case, we are forcing the city to be searched in Portugal). Figure 14 shows the application being executed.

5 Conclusion

In this article we introduced a tool that, by inspecting an XML file, specifies a set of object definitions in Perl, that can be used to represent the same data structure and semantics that are present in the original document. This set of objects can be later used in applications to reach and manipulate the data gathered or received via XML.
use Geo::Location;
my $place = shift;
my $url = "http://api.wunderground.com/auto/wui/geo/"
  . "GeoLookupXML/index.xml?query=$place, Portugal";
my $loc = Location->new({url => $url});
print "LAT $loc->lat. " LON "$loc->lon."\n";

Fig. 13. Code to query the geographic location web-service and print a city latitude and longitude using the object interface as created by xml2pm.

$ perl latlon.pl braga
LAT 41.58666611 LON -8.45666695

$ perl latlon.pl porto
LAT 41.22999954 LON -8.6800031

$ perl latlon.pl 'vila do conde'
LAT 41.34999847 LON -8.7500000

Fig. 14. Output from the geographic location web-service client.

The object itself is able of mirroring the information contained in the XML in his own attributes. The object definition by itself also provides an additional layer that can be used to perform common tasks related to data transformation.

One major drawback of this type of approach are the memory issues that can make this code unusable for big documents, as we are creating a tree of objects mostly like a Document Object Model parser would create. The main difference is that our approach creates code that is specifically created for this type of document.

Also, the fact that this code is generated during programming time means that the programmer can change the behavior of some of the methods, in order to obtain some specific validations or data conversions. Therefore, the created code can be more versatile than a simple generic API to convert XML to and from Objects.

This is a common and systematic approach when implementing applications that deal with data stored in XML format, therefore the use of a automatic tool that can perform most of the work for us with the least information available up front. It has proven useful and allows the implementation of applications much faster and in a elegant and modular way, since all the XML related code is delegated to the object itself. We have demonstrated this situation by showing how to implement a couple of simple example applications described earlier.

6 Future Work

This work is still under heavy development. Therefore, we have a big pile of features we would like to implement. The more imperious tasks that need to be addressed are:
Currently, the object definition is created only by inspecting an instance of an XML source file. The use of Document Type Definition (DTD) documents, or XML Schema documents as bootstrapping source could result in more generic code (we are not dealing with a specific instance) and we can gather some extra information that can be coded (for instance, checking values in accessors methods).

- Some recursive structures will imply the creation of several objects nested in each other. In the current version we are not fully addressing this issue. This situation needs to be well defined to allow the use of this tool in more complex case studies.

A more natural approach for handling elements’ attributes without changing the default accessor/setter behavior could also improve the overall quality of the generated code.

Acknowledgments

This work was partly supported by project CROSS (PTDC/EIA-CCO/108995/2008), funded by the Portuguese Foundation for Science and Technology.

References

Abstract. Data warehouses are used in many application domains, and there is no established method for their preservation. A data warehouse is structured by star or snowflake representations and can be grouped into data marts. A star is made up of a fact table that stores the facts, and dimensional tables that contextualize the facts. There are also bridge tables used to resolve a many to many relationship between a fact table and a dimension table, or to flatten out a hierarchy in a dimension table. A snowflake is similar to a star but where the dimension tables have suffered a partial normalization, resulting in subdimensions. A data warehouse can be implemented in multidimensional structures or relational databases that represents the dimensional model concepts in the relational model. The focus of this work is on describing the dimensional model of a data warehouse and migrating it to an XML model, in order to achieve a long-term preservation format. This paper presents the definition of the XML structure that extends the SIARD format used for the description and archive of relational databases, enriching it with a layer of metadata for the data warehouse components. Data Warehouse Extensible Markup Language (DWXML) is the XML dialect proposed to describe the data warehouse. To acquire the relevant metadata for the warehouse and build the archive format, an application was produced that combines the SIARD format and the DWXML metadata layer.

Keywords: Database Preservation, DWXML, SIARD format

1 Introduction

The technological generation in which we live has gradually modified the method to create, process and store information, using compulsively digital means for this purpose. The institutions, enterprises and governments rely more and more
on information systems that increase the availability and accessibility of information. These information systems typically require relational databases, transforming them into valuable assets for those entities.

However, rapid technological changes degenerate into rapid obsolescence of applications, file formats, media storage and even databases management systems (DBMS) [1]. If nothing is done, access to large chunks of stored information may become impossible and it be lost forever. So, it is important that entities which have major responsibilities in preserving information in digital form, become aware of this problem and join to initiatives all over the world, seeking for the best methodology for digital long-term preservation, and in particular for database preservation.

The present work is a development product of the DBPreserve project, a research project funded by the Portuguese Foundation for Science and Technology (FCT), in collaboration with INESC Porto, University of Minho and National Archives of Portugal (DGRARQ), aiming at studying the feasibility of using data warehousing technologies to preserve complex electronic records, such as those constituting databases. DBPreserve project approaches the long-term preservation of relational databases issue with a new concept, a two step migration:

- A model migration from the relational model to the dimensional model, using data warehouse concepts for model simplification and efficiency increase [2];
- An XML migration from the dimensional model to an XML [3] format that represents the data warehouse, to ensure a long-term preservation format.

A data warehouse has star or snowflake representation, made up of fact tables and dimensional tables that adds context and meaning to the facts. When a dimension table is partially normalized, resulting in subdimensions, it is called a snowflake schema. A bridge table is used between a fact table and a dimension table or to flatten out a hierarchy in a dimension table. Data marts are subsets of a data warehouse.

Data Warehouse Extensible Markup Language (DWXML) is an XML dialect with the purpose of describing a Data Warehouse (DW) [1, 4, 5]. It has been defined and refined according to data warehouse’s properties and tested using a case study of SiFEUP [2]. Its use in the project lies as a complement to the SIARD format [6] used for the description and archive of relational databases. This enrichment leverages past efforts to define an archive format suitable for data tables from databases and adds a layer of metadata for the data warehouse perspective.

2 Data Warehouse Preservation

Digital preservation has become more and more the focus for researching about what is the best strategy that is sustainable and efficient for the long-term preservation of digital objects [7]. Thibodeau’s organization of digital preservation strategies relate them to their applicability and objective [8].

1 http://www.fe.up.pt/a1/PROJECTOS_GERAL_MOSTRA_PROJECTO?ID=1349
2 Information System of Faculty of Engineering, University of Porto, Portugal
The Open Archival Information System (OAIS) Reference Model [9] introduces the appropriate terminology in the context of long-term preservation and defines the functional components necessary to implement an archive.

There are already many efforts and projects developed under the digital preservation scope. Projects such as CAMILEON [10], InterPARES [11], FEDORA [12] or PLANETS [13,14,16] contributed to the study of requirements, strategies and proposals for preserving digital objects and ensure their authenticity.

Regarding complex digital objects, such as databases, projects like SIARD [6], Chronos [17] or RODA [18], analyzed in detail the preservation of relational databases. PLANETS project built a framework that also deals with Access, MS SQL Server and Oracle databases, as well as the SIARD format [19].

Data warehouses are often implemented using relational database technology, and thus they are made up of tables that store data. A deeper inspection leads to the finding of facts, dimensions, bridges tables, indexes, level keys and views. However, there are some key differences between a database used in an operational system and in a data warehouse.

W. H. Inmon defined a data warehouse as “a subject-oriented, integrated, nonvolatile, time variant collection of data in support of management decisions” [4]. Data warehouses fulfill two major purposes: provide a single, clean and consistent source of data for decision support and unlink the decision platform from the operational system [1].

In a data warehouse the tables and joins are simple and de-normalized, in order to reduce the response time for analytical queries. For the characterization of a data warehouse additional metadata is required that defines the dimensional model and allows the data interpretation across different perspectives.

2.1 Data Warehouse Metadata

The structure of a data warehouse is referred to as a dimensional schema, where the fact tables are surrounded by dimensional tables, forming star schemas. A fact table is often located at the center of a star schema and consists of facts of a business process (e.g., measurements, metrics).

To understand the facts it is necessary to introduce the context and meaning of the dimensional model, achieved by the dimensions, representing the relevant vectors of analysis of the business process facts. The dimensions allow us to identify the how, what, who, when, where and why of something. Dimensions are usually represented by one or more dimensional tables. A dimensional table contains attributes in order to define and group the data for data warehouse querying.

The dimensions are characterized by a set of levels with defined hierarchies. Hierarchies are logical structures that use levels to organize and aggregate data, define navigation paths or establish a family structure [4,5]. A common example is a time dimension, a hierarchy might aggregate data from the day level to the week level to the month level to the quarter level to the year level.
The figure 1 shows an example of a star schema related to a real world case study used in the project, a “Course Evaluation System”, aiming to obtain general statistics about user satisfaction (anonymous students) in an academic environment scope, specifically on professor and class evaluation.

![Fig. 1. Star schema example](image)

In the center, a fact table contains the submitted answers (IPDW_ANSWERS). As dimensional tables, there are the question table (IPDW_QUESTION), the quiz table (IPDW_QUIZ), also the semester table (IPDW_SEMESTER), the class table (IPDW_CLASS) and the professor table (IPDW_PROFESSOR). Because the answers are anonymous, there is no relationship towards the students, who actually answered the questions. An important step in the data warehouse building process is to declare the dimensions. The next sample code shows the declaration of a dimension with the CREATE DIMENSION SQL statement [20] using Oracle.

**Example of a dimension declaration**

```
CREATE DIMENSION class_dim
  LEVEL class IS (IPDW_CLASS.CLASS_ID)
  LEVEL course IS (IPDW_CLASS.COURSE_ID)
  HIERARCHY class_rollup(  
    class CHILD OF course)
  ATTRIBUTE class DETERMINES  
    (IPDW_CLASS.CURR_CODE, IPDW_CLASS.CURR_ACRONYM,  
     IPDW_CLASS.CURR_NAME, IPDW_CLASS.CURR_TYPE)
  ATTRIBUTE course DETERMINES  
    (IPDW_CLASS.CURR_CODE, IPDW_CLASS.CURR_ACRONYM,  
     IPDW_CLASS.CURR_NAME, IPDW_CLASS.CURR_TYPE,  
     IPDW_CLASS.CURR_PREVIOUS_CDD)
```

This declaration defines a dimension (class_dim) with a hierarchy (class_rollup) of two levels: the level course with COURSE_ID as level key, and a child level class with CLASS_ID as level key. This dimension uses the data from the table IPDW_CLASS. The ATTRIBUTE clause specifies the attributes that are uniquely determined by a hierarchy level. Thus it is possible to analyze the data in a more global perspective, through the course level, or get a more detailed overview using the class level.
Another data warehouse concept is a bridge table. A bridge table is used to resolve a many to many relationship between a fact table and a dimension table and is also used to flatten out a hierarchy in a dimension table [5].

Storing snowflake schemas and data marts is also needed. The snowflake schema is similar to the star schema, but dimensions are normalized into multiple related tables. A data mart is a subset of a data warehouse [5, 21].

2.2 Data Warehouse Preservation Format Proposal

The main objective of this study was to obtain a preservation format that suited the characteristics of a generic data warehouse. This format should allow the definition of the relevant metadata from the perspective of the data warehouse and archive the relevant metadata as well as the data from the tables in a format that would guarantee long-term preservation. The use of XML to the verification of these requirements appeared as the next option.

The study of the work already produced around the preservation of databases [6, 17, 18], including the model migration approach developed in the DBPreserve project [2], and on XML representation of a data warehouse [22, 23], resulted in the decision to complement the SIARD format, an XML based format for the archival of relational databases, in order to adapt it to the characteristics of the dimensional model used in data warehouses.

The SIARD format proved to be the most appropriate starting point for this representation given the inherent modularity of data warehouses, with independent stars sharing some dimensions. SIARD has a segmented structure of directories and files, unlike DBML [18] (Database Markup Language) presented at RODA, which represents everything in a single file, impairing the handling of data.

Thus, reusing the effort to define an archive format that stores the definition of the tables and their data, it is proposed to add a metadata layer for data interpretation according to the data warehouse perspective. So, given the simplicity of the dimensional model in terms of relationships between tables, it becomes possible to analyze the archived data with greater efficiency through simplified queries applied directly on the XML files using XQuery\(^3\) and XPath\(^4\).

3 Relational Database Preservation with SIARD

The Swiss Federal Archives (SFA) have developed an open storage format for relational databases called SIARD\(^5\) (Software Independent Archiving of Relational Databases), as well as a set of conversion tools named the SIARD Suite [24], in order to convert relational databases (e.g., Access, Oracle and SQL Server) into the archival SIARD format, edit the SIARD format and reactivate an archived database, restoring from the SIARD Format to a database.

\(^3\) [http://www.w3.org/TR/xquery](http://www.w3.org/TR/xquery)

\(^4\) [http://www.w3.org/TR/xpath](http://www.w3.org/TR/xpath)

\(^5\) Official site: [http://www.bar.admin.ch](http://www.bar.admin.ch)
The SIARD format is a nonproprietary and published open standard, based on open standard (e.g., ISO norms Unicode, XML, SQL1999) and the industry standard ZIP. In May 2008, the European PLANETS project accepted SIARD format as the official format for archiving relational databases [6].

The SIARD format is a ZIP64 [25] uncompressed package based on an organizational system of folders, storing the metadata in the header folder and table data in the content folder. This organization is shown in figure 2.

For database’s metadata characterization a single XML file is used that contains the entire structure of the database (schemas, tables, attributes, keys, views, functions...) and the corresponding XSD\(^6\) schema for XML validation.

As to the primary data, each schema is stored in different folders and sequentially numbered, as well as the tables of each schema. The data from each table is stored in an XML file with simplified structure (only rows and columns) and its XSD. If there are Large Objects - LOB (BLOB - Binary Large Objects and CLOB - Character Large Objects), these data are stored in binary files or text, within a folder for each attribute of these types, being referred to its path in the respective XML of the table.

3.1 SIARD Suite

The SIARD project produced a set of tools - SIARD Suite\(^7\) [24] - comprised of three components: the SiardEdit, a graphical user interface for migration and metadata processing; the SiardFromDb, a command line application for extracting and storing a database generating the SIARD file; and the SiardToDb, a command line application to reactivate a database from a SIARD file.

---

\(^{6}\) http://www.w3.org/XML/Schema

\(^{7}\) This application was gently sent by Johannes Bader from SIARD project
4 DWXML definition

Regarding the SIARD format extension for archiving data warehouses, the proposed XML bridges the gap to describe the dimensional model, adding a metadata file (dw.xml) and its schema definition (dw.xsd\(^8\)). The figure 3 shows an excerpt of the extended SIARD format, bearing the description of a data warehouse.

![Fig. 3. DWXML added to the SIARD Archive File](image)

This study characterizes the data warehouse as a set of stars and a set of dimensions, represented in tables and views organized in schemas. It is also envisaged a representation of data marts. The figure 4 caracterizes the DWXML basic structure and the star element.

The schemas, tables and views follow a similar representation to the SIARD format and are replicated in this description to permit the characterization of a data warehouse regardless of whether there is or not a package SIARD. However, this DWXML version does not contemplate the representation of the primary data in XML, since it is used in conjunction with the format SIARD, which already performs the primary data migration to XML format.

The attribute version represents the version of the DWXML definition. The dwBinding element supports the description of the DWXML file, the information related to the owner of the data, the credentials of the connection to the data warehouse and the names and versions of the applications involved in the DWXML creation, including the DBMS where the data warehouse was working.

4.1 Stars and Facts

A star is composed of a fact table and a set of rays which establish relationships to dimensions and possibly bridge tables. The factTable element references the respective table description in the schemas element, it indicates the columns responsible for the joins between fact tables and bridge tables or dimensions, it contains information about its granularity and about the facts. With respect to

\(^8\) [https://www.fe.up.pt/siwikis_paginas_geral.paginas_view?pct_pagina=42633](https://www.fe.up.pt/siwikis_paginas_geral.paginas_view?pct_pagina=42633)
the facts, they indicate the table column that represents them, as well as their measure type: non-additive, semi-additive or additive.

In a star, each ray element represents a relationship between the fact table and the dimension. If there is a many to many relationship between the fact table and the dimension table, it could be added up a bridge table. In this case, the ray element would be compose by a bridgeTable element that references the related table, followed by the dimension element that represents a reference to the dimension.

**Example of a DWXML star definition**

```xml
<?xml version="1.0" encoding="UTF-8"?>
<dwxml version="1.0" xsi:noNamespaceSchemaLocation="dw.xsd"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<stars>
<star>
  <name>IPDW_ANSWERS_STAR</name>
  <description>Star related to the answers</description>
  <schema>CALDEIAS</schema>
  <name>IPDW_ANSWERS</name>
  <fact>
    <name>ANSWER</name>
    <column>ANSWER</column>
    <measure>ADDITIVE</measure>
  </fact>
</star>
</stars>
</dwxml>
```
4.2 Dimensions

A key step in the process of the data warehouse creation is to declare the dimensions [20], so that the data dictionary [26] contains this metadata and enables its future extraction. It eases the process of identifying the dimensions, levels and hierarchies, as well as tables and views that support them. The figure 5 displays the dimensions element schema.

![Dimensions element schema](image)

Fig. 5. The dimensions element schema

The metadata related to the dimensions is stored in separated dimension elements and allows the categorization and description of the facts and measures in order to support meaningful answers to the requested questions. Each dimension element describes the levels and respective level keys, the level hierarchies and the attributes defined by each level. The tables and views elements contain the reference to the tables and views described in the schemas element.

5 Application Architecture

The DBPreserve Suite, the application that supports the data warehouse migration process to the proposed preservation format, has the following general
requirements: to get the metadata describing the data warehouse, to integrate the component SiardFromDb that migrates the data warehouse to the SIARD format, to generate the DWXML and add it to the generated SIARD file and must have a graphical interface that helps the migration process and allows editing and retrieving of metadata by querying the primary data in XML format.

This application is composed by 5 major modules as shown in the overall architecture of the application in figure 6 and it has been developed using the NetBeans IDE 7.0 RCl and Netbeans Platform\(^9\), with support for Java 1.7\(^{10}\), using the JDOM\(^{11}\) library [27] for XML processing. The DBPreserve Suite has been tested in a case study that uses a data warehouse built on Oracle Database 11g Enterprise Edition Release 11.1.0.7.0 - 64bit Production\(^{12}\).

![DBPreserve Suite general architecture](image)

Fig. 6. DBPreserve Suite general architecture

The metadata extraction needed to complete the DWXML is done using a module that requests the metadata from the data dictionary \([26]\) of the data warehouse. Through the analysis of the acquired metadata, a significant part of the metadata is automatically filled in, directly or by inference. Nevertheless, it is still necessary some manual input of small metadata details, such as objects’ descriptions.

The SIARD Suite component that builds the SIARD format is integrated into the DBPreserve Suite via a thread responsible for the process that manages the execution of the command SiardFromDb \([24]\), as well as the log of the migration execution. At this stage, the object to migrate is the relational database implementation of a data warehouse.

For the SIARD format encapsulation, the SIARD Suite uses a proprietary format to create the uncompressed ZIP64, that extends the ZIP format to overcome the 4 GB size limit in the standard ZIP. However, the access and the integration of DWXML into the SIARD format is performed using the Java 1.7 java.util.zip library which already supports ZIP64 format extensions defined by the PKWARE ZIP File Format Specification \([25]\).

\(^9\) [http://netbeans.org/features/platform/](http://netbeans.org/features/platform/)

\(^10\) [http://download.java.net/jdk7/docs/api/](http://download.java.net/jdk7/docs/api/)

\(^11\) [http://www.jdom.org/index.html](http://www.jdom.org/index.html)

The DWXML generation is performed by a Java representation of an XML document using JDOM [27]. JDOM has a straightforward, fast and lightweight API, optimized for Java programming.

The output module enables the access and display of the XML archived data throughout the data warehouse perspective and allows star level queries, using XQuery and XPath.

6 Conclusions and Future Work

This study resulted in a proposed file format for long-term preservation of data warehouses. The DWXML presented allows the characterization of the data warehouse metadata and seamlessly extends the SIARD format for this kind of databases. The developed application allows the control over the process of migrating the data warehouse and associated metadata to XML, according to DWXML and SIARD Format, as well as adding and editing associated metadata. Since this is an XML archive from a dimensional model, with simplified relationships, it is possible to query and extract the stored data with higher performance rather than using an XML from a relational model. As future work, there is the intention of untying the application from the SIARD Suite that makes the migration of primary data in the SIARD format with heavy costs in terms of time consumption, testing the performance improvements introduced by Java 1.7 and the use of JDOM in the XML processing. Another contribute to the enrichment of this application can be the reactivation of the data warehouse in a DBMS, in order to restore the data warehouse from the XML based archive format described.

References


Using Ontologies in Database Preservation

Ricardo André Pereira Freitas
José Carlos Ramalho

1 CLEGI – Lusíada University
Vila Nova de Famalicão – Portugal
2 Department of Informatics – University of Minho
Braga – Portugal
freitas@famulusiada.pt, jcr@di.uminho.pt

Abstract. This paper addresses the problematic Digital Preservation and focuses on the conceptual model within a specific class of digital objects: Relational Databases. Previously, a neutral format was adopted to pursue the goal of platform independence and to achieve a standard format in the digital preservation of relational databases, both data and structure (logical model). Currently, in this project, we intend to address the preservation of relational databases by focusing on the conceptual model of the database, considering the database semantics as an important preservation “property”. For the representation of this higher level of abstraction present in databases we use an ontology based approach. At this higher abstraction level exists inherent knowledge associated to the database semantics that we tentatively represent using “Web Ontology Language” (OWL). We developed a prototype (supported by case study) and define a mapping algorithm for the conversion between the database and OWL. The ontology approach is adopted to formalize the knowledge associated to the conceptual model of the database and also a methodology to create an abstract representation of it.

Key words: Digital Preservation, Relational Databases, Ontology, Conceptual Models, Knowledge, XML, Digital Objects

1 Introduction

In the current paradigm of information society more than one hundred exabytes of data are used to support information systems worldwide [1]. The evolution of the hardware and software industry causes that progressively more of the intellectual and business information are stored in computer platforms. The main issue lies exactly within these platforms. If in the past there was no need of mediators to understand the analogical artifacts today, in order to understand digital objects, we depend on those mediators (computer platforms).

Our work addresses this issue of Digital Preservation and focuses on a specific class of digital objects: Relational Databases (RDBs). These kinds of archives are important to several organizations (they can justify their activities and characterize the organization itself) and are virtually in the base of all dynamic content in the Web.
1.1 Previous Works

In previous work [2] we adopted an approach that combines two strategies and uses a third technique — migration and normalization with refreshment:

- Migration which is carried in order to transform the original database into the new format – Database Markup Language (DBML) [3];
- Normalization reduces the preservation spectrum to only one format;
- Refreshment consists on ensuring that the archive is using media appropriate to the hardware in usage throughout preservation [4].

This previous approach deals with the preservation of the Data and Structure of the database, i.e., the preservation of the database logical model. We developed a prototype that separates the data from its specific database management environment (DBMS). The prototype follows the Open Archival Information System (OAIS) [5] reference model and uses DBML neutral format for the representation of both data and structure (schema) of the database.

1.2 Conceptual Preservation

In this paper, we address the preservation of relational databases by focusing on the conceptual model of the database (the information system - IS). For the representation of this higher level of abstraction present in databases we use an ontology based approach. At this level there is an inherent Knowledge associated to the database semantics that we represent using OWL [6].

We developed a prototype (supported by case study) and established an algorithm that enables the mapping process between the database and OWL.

In the following section, we overview the problem of digital preservation, referring to the digital object and preservation strategies. The next section also formulates our hypothesis. In section 3 we overview the relation between ontologies and databases. The prototype and the mapping process from RDBs to OWL is detailed in section 4. At the end we draw some conclusions and specify some of the future work.

2 Digital Preservation

A set of processes or activities that take place in order to preserve a certain object (digital) addressing its relevant properties, is one of the several definitions. Digital objects have several associated aspects (characteristics or properties) that we should consider whether or not to preserve. The designate community plays an important role and helps to define

"The characteristics of digital objects that must be preserved over time in order to ensure the continued accessibility, usability, and meaning of the objects, and their capacity to be accepted as evidence of what they purport to record" [7].
2.1 The Digital Object

Some distinction can be established between digital objects that already born in a digital context, and those that appear from the process of digitization: analog to digital. In a comprehensive way and encompassing both cases above, we can consider that a digital object is characterized by being represented by a bitstream, i.e., by a sequence of binary digits (zeros and ones) [8].

We can question if the physical structure (original system) of the object is important, and if so, think about possible strategies for preservation at that level (museums of technology). Nevertheless, the next layer — the logical structure or logical object—, which corresponds to the string of binary digits will have different preservation strategies. The bitstream have a certain distribution that will define the format of the object, depending on the software that will interpret it. The interpretation by the software, of the logical object, provides the appearance of the conceptual object, that the human being is able to understand (interpret) and experiment. The strategy of preservation is related to the level of abstraction considered important for the preservation [9]. From a human perspective one can say that what is important to preserve is the conceptual object (the one that the humans are able to interpret). Other strategies defend that what should be preserved is the original bitstream (logical object) or even the original media. Figure 1 shows the relationship between the different levels of abstraction (digital object) and the correspond preservation formats adopted for RDBs in this research.

![Diagram of levels of abstraction and preservation policy](image)

**Fig. 1. Levels of Abstraction and Preservation Policy**

By focusing on a specific class or family of digital objects (relational databases), questions emerge such as: what are the effects of cutting/extracting the object from its original context? Can we do this even when we are referring to objects that are platform (hardware/software) dependent? The interaction between the source of the digital object and the platform results on a conceptual object that
can be different if the platform changes [7]; the output can be different (will the object maintain its original behavior?). The important is the preservation of the essential parts that purport what the object where made for. Either the source or the platform can be altered if what is essential is obtained and also maintaining the meaning of the digital object over long periods of time (long-term scope).

As we mentioned in previous work we address the preservation of the RDBs data and structure by using DBML which ensures that its representation becomes neutral.

2.2 Proposed Approach

Our hypothesis concentrates on the potentiality of reaching relevant stages of preservation by using ontologies to preserve of RDBs. This lead us to the preservation of the higher abstraction level present in the digital object, which corresponds to the database conceptual model. At this level there is an inherent Knowledge associated to the database semantics (Table 1). We intend to capture the experimented object (knowledge) through an ontology based approach. The ontology approach is adopted to formalize the knowledge present at the experimented object level and also a methodology to create an abstract representation of it.

<table>
<thead>
<tr>
<th>Digital Object</th>
<th>Preservation Levels</th>
<th>Relational Database</th>
</tr>
</thead>
<tbody>
<tr>
<td>Experimented Object</td>
<td>Ontology</td>
<td>Conceptual Model</td>
</tr>
<tr>
<td>Conceptual Object</td>
<td>DBML</td>
<td>Logical Model</td>
</tr>
<tr>
<td>Logical Object</td>
<td>–</td>
<td>Original Bitstream</td>
</tr>
<tr>
<td>Physical Object</td>
<td>–</td>
<td>Physical Media</td>
</tr>
</tbody>
</table>

3 Ontologies and Relational Databases

There is a direct relation between ontologies and databases: a database has a defined scope and intends to model reality within that domain for computing (even when it is only virtual or on the web); ontology in ancient and philosophical significance means the study of being, of what exists [10].

The (strong) entities present in relational databases have an existence because they were model from the real world; they relate to each other and have associated attributes. In information society and computer science, an ontology establishes concepts, their properties and the relationships among them within a given domain [10].
3.1 Ontologies

The study of ontologies in computer science received new impetus due to the growth of the web, their associated semantics and the possibility of extracting knowledge from it. The "Semantic Web" supported by W3C works on establishing a technology to support the Web of data [11]. Notice that a tremendous part of the web is based on (relational) databases - specially dynamic information. An ontology can provide readable information to machines [12] at a conceptual level (higher abstraction level). They also enable the integration and interpretability of data/information between applications and platforms.

3.2 Database Semantics

A database can be defined as a structured set of information. In computing, a database is supported by a particular program or software, usually called the Database Management System (DBMS), which handles the storage and management of the data. In its essence a database involves the existence of a set of records of data. Normally these records give support to the organization information system; either at an operational (transactions) level or at other levels (decision support - data warehousing systems).

If we intend not only to preserve the data but also the structure of the (organization) information system we should endorse efforts to characterize (read) the database semantics. In other words, we represent the conceptual model of the database using an ontology and intend to preserve that representation.

Ontologies benefit from the fact that they are not platform/system dependent when compared to traditional relational databases.

4 From RDBs to OWL

This section presents the work developed to convert databases to ontology, based on a mapping process (mapping algorithm), for preservation. We intend to preserve a snapshot of the database (or a frozen database) by preserving the OWL generated from the database.

We start to briefly refer to some of the related work in this area considering the numerous approaches addressing conversions and mappings between relational databases and ontologies. Then we concentrate our efforts on detailing the mapping process and analyzing the created algorithm. The conducted tests and some of the results are also presented.

4.1 Related Work

Several approaches concerning RDBs and ontologies transformations exist and are being addressed continuously. The conversion from databases into an ontology could be characterized as a process in the scope of reverse engineering [13]. While some approaches and works try to establish a mapping language or
a mapping process [14], others use different techniques and strategies for the database translation [15] into an ontology (e.g., OWL).

Considering the Resource Description Framework (RDF) [16] and RDBs, some of the related works, studies and tools are referenced in the W3C Incubator group survey [17]: Virtuoso RDF View; D2RQ; Triplify; R2O; Dartgrid Semantic Web toolkit; RDBToOnto, and others. The extraction of ontologies from RDBs are also addressed and referenced in [12].

### 4.2 Mapping Process of RDBs to OWL – Prototype

Our work implements the conversion from RDBs into OWL through an algorithm that performs the mapping process. The developed prototype enables the connection to a DSN (Data Source Name), extracts the data/information needed and gives the initial possibility of selecting the tables of interest (for conversion). It is assumed that the source database is normalized (3NF).

Let’s start by enumerating the properties of RDBs that are address and incorporated in the ontology (OWL):

- **Tables** names;
- **Attributes** names and data types;
- **Keys** primary keys, foreign keys (relationships between tables);
- **Tuples** data;

These elements are extracted from the database into multidimensional arrays. Figure 2 shows the arrays structure.

![Fig. 2. Multidimensional Array Structure](image)

For each **table** on the database we define a **class** on the ontology with the exception of those tables where all attributes constitute a composed primary key (combination of foreign keys). These link tables used in the relational model to
dismount a many-to-many relationship, are not mapped to OWL classes, instead they give origin to **object properties** in the ontology. These object properties have on there domain and range the correspondent classes (database tables) involved in the relationship (Fig. 3).

![Algorithm - Classes and Non Classes](image)

The **foreign keys** of the tables mapped directly to OWL classes also give origin to **object properties** of the correspondent OWL classes (tables). The **attributes** of the several tables are mapped to **data properties** within the analogous OWL classes with the exception of the attributes that are foreign keys (Fig. 4).

The algorithm generates inverse object properties for all relationships among the classes. If the object properties are generated directly from a 1-to-many relationship (which is the last case) it is possible to define one of the object properties as functional (in one direction).

The **tuples** of the different tables are mapped to **individuals** in the ontology and are identified by the associated **primary key** in the database. A tuple in a database table is mapped to an individual of a class (Fig. 5).

The object properties that relates individuals in different classes are only defined in one direction. If in the inverse pair of object properties exists one property that is functional, is that one that it is defined; if not, the generated object property assertion is irrelevant.

In the next table (Fig. 6) we summarize the mapping process. From the conceptual mapping approach and some DBMS heuristics we start to manually convert a relational database (case study database) into OWL using Protégé [18]. The algorithm was then designed based on the defined mapping and from the code analysis (Protégé – OWL/XML format).

### 4.3 Prototype – Tests and Results

The algorithm was then tested with the case study database. Figure 7 shows the database logical model and the ontology conceptual approach. It was nec-
Fig. 4. Algorithm - Structure Generation

Fig. 5. Algorithm - Individuals

Fig. 6. Mapping Process Sumarized
It is necessary to do some adjustments in order to achieve a consistent ontology. Then we successfully use the HermiT 1.3.3 reasoner [19] to classify the ontology. The inverse "object properties assertions" that the algorithm do not generates for the individuals were inferred. Some equivalent (and inverse functionality) object properties were also inferred.

Fig. 7. RDB Logical Model vs Ontology Overview

The next step consisted on testing the algorithm with other databases. We use one MySQL database and two MSSQL Server databases (the maximum tables size were about tens of thousands records). All databases used in this research are from the University Lusiada information system.

The results were very satisfactory because the algorithm achieve similar results of the ones obtained with the case study database only with minor inconsistencies. The processing time is an issue directly related to the dimension of the database (it is necessary to test the algorithm with huge databases [millions of records] in machines with powerful processing capability).

5 Conclusion and Future Work

Ontologies and databases are related to each other because of their characteristics. Using ontologies in database preservation is an approach to capture the "knowledge" associated to the conceptual model of the database.

In previous work we preserve the database data and structure (logical model) by ingesting the database in a XML based format into an OAIS based archive.

Here, we present the work developed in order to convert databases to ontology, based on a mapping process (mapping algorithm), for preservation. In order to preserve a snapshot of the database (or a frozen database) we preserve the ontology (OWL, also a XML based format) obtained from the application of the developed algorithm to the source database. We tested the algorithm with few
databases and the results were acceptable in terms of consistency of generated ontology (and comparing to the results obtained with the case study database).

This generated ontologies will induce the development of a new database browser/navigation tool.

Ontologies also have other potentialities such as the asset of providing answers to questions that other standards are limited. For example, in terms of metadata, one issue that we intend to also address in future work.

We also anticipate the possibility of integration between OWL Web Ontology Language [6] and Semantic Web Rule Language (SWRL) [20] to consolidate the asserted and inferred knowledge about the database and its information system.

References

11. http://www.w3.org/standards/semanticweb/
## Author Index

<table>
<thead>
<tr>
<th>Name</th>
<th>Page(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alberto Simões</td>
<td>59, 102</td>
</tr>
<tr>
<td>Alexandre B. Augusto</td>
<td>23</td>
</tr>
<tr>
<td>Carlos Aldeias</td>
<td>115</td>
</tr>
<tr>
<td>Cristina Ribeiro</td>
<td>83, 115</td>
</tr>
<tr>
<td>Gabriel David</td>
<td>115</td>
</tr>
<tr>
<td>João Correia Lopes</td>
<td>83</td>
</tr>
<tr>
<td>João Silva</td>
<td>83</td>
</tr>
<tr>
<td>Jorge Coelho</td>
<td>70</td>
</tr>
<tr>
<td>José Carlos Ramalho</td>
<td>127</td>
</tr>
<tr>
<td>José João Almeida</td>
<td>102</td>
</tr>
<tr>
<td>José Paulo Leal</td>
<td>37, 91</td>
</tr>
<tr>
<td>Liliana Alexandre</td>
<td>70</td>
</tr>
<tr>
<td>Mário Pinto</td>
<td>49</td>
</tr>
<tr>
<td>Manuel E. Correia</td>
<td>23</td>
</tr>
<tr>
<td>Nuno Carvalho</td>
<td>102</td>
</tr>
<tr>
<td>Pedro Assis</td>
<td>3</td>
</tr>
<tr>
<td>Ricardo Freitas</td>
<td>127</td>
</tr>
<tr>
<td>Ricardo Queirós</td>
<td>37, 49, 91</td>
</tr>
<tr>
<td>Sara Fernandes</td>
<td>59</td>
</tr>
<tr>
<td>Tiago Andrade e Silva</td>
<td>19</td>
</tr>
</tbody>
</table>