Universidade de Aveiro Departamento de Electrónica, Telecomunicações e Informática
2011-2012

Pedro Jorge Pereira Lopes

Composição de Serviços para Aplicações Biomédicas

Service Composition for Biomedical Applications

Universidade de Aveiro Departamento de Electrónica, Telecomunicações e Informática
2011-2012

Pedro Jorge Pereira Lopes

Composição de Serviços para Aplicações Biomédicas

Service Composition for Biomedical Applications

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Doutor em Engenharia Informática, realizada sob a orientação científica do Doutor José Luís Guimarães Oliveira, Professor Associado do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro

o júri

presidente

Doutor Artur Manuel Soares da Silva
Professor Catedrático do Departamento de Química, da Universidade de Aveiro

Doutor Víctor Maojo García
Professor Catedrático Faculdade de Informática, da Universidade Politécnica de Madrid

Doutor Rui Pedro Sanches de Castro Lopes
Professor Coordenador do Departamento de Informática e Comunicação, da Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Bragança

Doutor Francisco José Moreira Couto
Professor Auxiliar do Departamento de Informática da Faculdade de Ciências, da Universidade de Lisboa

Doutor Carlos Manuel Azevedo Costa
Professor Auxiliar do Departamento de Electrónica, Telecomunicações e Informática, da Universidade de Aveiro

Doutor José Luís Guimarães Oliveira
Professor Associado do Departamento de Electrónica, Telecomunicações e Informática, da Universidade de Aveiro

agradecimentos

First of all I would like to thank my supervisor José Luís Oliveira for taking me on as a student during these five, long (and yet short) years, and for sticking with me through the good times and the bad, right until the end of the line. As I have learned a great deal from you, I hope you have learned perhaps a little bit from me too.
I also want to thank everybody in the UA.PT Bioinformatics team at IEETA, who were with me on much of this journey. You have provided support to my science through stimulating discussions and helped my mental sanity with all other topics, from the best TV shows to what features will be available in the next Apple product.
My deepest acknowledgement goes to my tireless Mother, whose support throughout these years has been tremendously important to me at all levels. You are, and will always be, my inspiration and role model.

Last, but not least, I want to thank my girlfriend Andreia, for everything and so much more.

palavras-chave

Composição de serviços, aplicações biomédicas, bioinformática, sistemas de informação, workflow, web semântica.

resumo

A exigente inovação na área das aplicações biomédicas tem guiado a evolução das tecnologias de informação nas últimas décadas. Os desafios associados a uma gestão, integração, análise e interpretação eficientes dos dados provenientes das mais modernas tecnologias de hardware e software requerem um esforço concertado. Desde hardware para sequenciação de genes a registos electrónicos de paciente, passando por pesquisa de fármacos, a possibilidade de explorar com precisão os dados destes ambientes é vital para a compreensão da saúde humana. Esta tese engloba a discussão e o desenvolvimento de melhores estratégias informáticas para ultrapassar estes desafios, principalmente no contexto da composição de serviços, incluindo técnicas flexíveis de integração de dados, como warehousing ou federação, e técnicas avançadas de interoperabilidade, como serviços web ou LinkedData.
A composição de serviços é apresentada como um ideal genérico, direcionado para a integração de dados e para a interoperabilidade de software. Relativamente a esta última, esta investigação debruçou-se sobre o campo da farmacovigilância, no contexto do projeto Europeu EU-ADR. As contribuições para este projeto, um novo standard de interoperabilidade e um motor de execução de workflows, sustentam a sucesso da EU-ADR Web Platform, uma plataforma para realizar estudos avançados de farmacovigilância. No contexto do projeto Europeu GEN2PHEN, esta investigação visou ultrapassar os desafios associados à integração de dados distribuídos e heterogéneos no campo do varíoma humano. Foi criada uma nova solução, WAVe - Web Analyses of the Variome, que fornece uma coleção rica de dados de variação genética através de uma interface Web inovadora e de uma API avançada. O desenvolvimento destas estratégias evidenciou duas oportunidades claras na área de software biomédico: melhorar o processo de implementação de software através do recurso a técnicas de desenvolvimento rápidas e aperfeiçoar a qualidade e disponibilidade dos dados através da adopção do paradigma de web semântica.
A plataforma COEUS atravessa as fronteiras de integração e interoperabilidade, fornecendo metodologias para a aquisição e tradução flexíveis de dados, bem como uma camada de serviços interoperáveis para explorar semanticamente os dados agregados. Combinando as técnicas de desenvolvimento rápidas com a riqueza da perspectiva "Semantic Web in a box", a plataforma COEUS é uma aproximação pioneira, permitindo o desenvolvimento da próxima geração de aplicações biomédicas.

keywords

Service composition, biomedical software, bioinformatics, information systems, workflow, semantic web.

abstract

The demand for innovation in the biomedical software domain has been an information technologies evolution driver over the last decades. The challenges associated with the effective management, integration, analyses and interpretation of the wealth of life sciences information stemming from modern hardware and software technologies require concerted efforts. From gene sequencing hardware to pharmacology research up to patient electronic health records, the ability to accurately explore data from these environments is vital to further improve our understanding of human health. This thesis encloses the discussion on building better informatics strategies to address these challenges, primarily in the context of service composition, including warehousing and federation strategies for resource integration, as well as web services or LinkedData for software interoperability.

Service composition is introduced as a general principle, geared towards data integration and software interoperability. Concerning the latter, this research covers the service composition requirements within the pharmacovigilance field, namely on the European EU-ADR project. The contributions to this area, the definition of a new interoperability standard and the creation of a new workflow-wrapping engine, are behind the successful construction of the EU-ADR Web Platform, a workspace for delivering advanced pharmacovigilance studies. In the context of the European GEN2PHEN project, this research tackles the challenges associated with the integration of heterogeneous and distributed data in the human variome field. For this matter, a new lightweight solution was created: WAVe, Web Analysis of the Variome, provides a rich collection of genetic variation data through an innovative portal and an advanced API. The development of the strategies underlying these products highlighted clear opportunities in the biomedical software field: enhancing the software implementation process with rapid application development approaches and improving the quality and availability of data with the adoption of the Semantic Web paradigm.

COEUS crosses the boundaries of integration and interoperability as it provides a framework for the flexible acquisition and translation of data into a semantic knowledge base, as well as a comprehensive set of interoperability services, from REST to LinkedData, to fully exploit gathered data semantically. By combining the lightness of rapid application development strategies with the richness of its "Semantic Web in a box" approach, COEUS is a pioneering framework to enhance the development of the next generation of biomedical applications.

“I think the biggest innovations of the twenty-first century will be the intersection of biology and technology.”

- Steve Jobs

TABLE OF CONTENTS

1. INTRODUCTION

"Words, so innocent and powerless as they are, as standing in a dictionary, how potent for good and evil they become in the hands of one who knows how to combine them."
- Nathaniel Hawthorne

Computer science has been evolving vertiginously since the middle of the 20th century. Converging with these phenomena, especially in the last decades, the life sciences research field fosters this innovation through a constant demand of newer and more advanced computational tools. As the "omics" revolution unfolds, sophisticated biomedical applications require best-of-breed technologies to cope with more complex requirements from miscellaneous niche fields within the life sciences.

Rather than being a simple auxiliary tool, bioinformatics software is essential for understanding disease aetiology. The completion of the draft human genome sequence [1, 2] promptly started a new genomics era [3]. Where traditional genetics research focused on analysing genes as individual entities, modern genomics envisages a grander comprehension of all connections from our DNA sequence - the *genotype* - to the observable changes in our organism - the *phenotype*. Understanding the genotype-to-phenotype connections is the cornerstone to understanding ourselves as humans, and to decipher the "Book of Life".

The genotype-to-phenotype domain plays a key role in future visions for individualized healthcare. With computer science advances, clinicians' strategies based on trial and error are being replaced by more precise treatments sustained by advanced software infrastructures. These approaches push forward the need for taking state-of-the-art technologies to a new level, through the better prediction, prevention, diagnostic and treatment of subtypes of diseases [4]. To achieve this, individualized healthcare must cross paths with custom drug design research and both will, in a long-term, realize the personalized medicine premise [5, 6].

It is up to computer science to deliver hardware and software to tackle the myriad of challenges emerging from the life sciences field. Genotype-to-phenotype investigations or pharmacology research involve high-throughput sequencing technologies and huge numbers of biological samples, generating data in immeasurable quantity and diversity. These data need to be evaluated, interpreted and integrated with other data to generate new knowledge [7]. To ensure the effective exploitation of this wealth of data, which results in truly valuable research, new service composition technologies and strategies are essential. Hence, the thrust of this thesis is the vital role played by service composition strategies in bioinformatics data integration and biomedical software interoperability.

The explosive evolution of biomedical and computational biology hardware and software technologies is making these fields the subject of extensive research projects. However, whereas in an initial stage, the adopted approaches consisted on the translation of state-of-the-art computer science technologies to the life sciences domain, nowadays, bioinformaticians demands are quickly surpassing available technologies, driving the development of more advanced systems. With these challenges in mind, this research endeavour seeks to bring innovation to the life sciences field once again, moving computer science technologies one-step ahead of biomedical applications' demands. From *ad hoc* service composition in biomedical applications to general-purpose service composition frameworks, research conducted in this doctorate covers the evaluation of existing tools and introduces newly developed solutions that are vital to keep bioinformatics at the boundaries of computer science innovation.

1.1 Thesis aims

The research conducted in this doctorate and detailed in this thesis is focused on the service composition, data integration and software interoperability challenges brought about by biomedical applications' evolution. The overall goals of this work are as follows.

- To **evaluate** service composition strategies as data integration and software interoperability enablers in the life sciences. Focus is given to the inward data flow - from external, distributed and heterogeneous resources - complemented by the outward data flow - advanced methods to publish data from a knowledge base. These address the general need for new approaches in the field that help facilitate information integration and exchange.
- To **explore** the potential role of state-of-the-art service composition technologies for biomedical applications. Particular areas include the tailored use of web services, the study of bioinformatics integration models and the assessment of strategies in place at widely used resources.

→ To **develop** new service composition strategies geared towards enhancing how data can be collected and addressing how it can be easily accessible later. Notwithstanding the vast amount of techniques and technologies available, emphasis is pointed to the semantic web as a streamlined development paradigm to acquire, disseminate and reason over knowledge.

These objectives can be summarized in an overarching research challenge: *What are the best state-of-the-art strategies for service composition in biomedical applications? How can these approaches be explored to improve existing scenarios? What are the strengths and opportunities that should be embraced to enhance resource integration and software interoperability in bioinformatics and computational biology?*

1.2 Contributions

With the biomedical software setting in the background, the research conducted in this doctorate originated a useful and advanced set of contributions towards both integration and interoperability.

Regarding interoperability, modern computer science technologies exploit the growing number of application programming interfaces to enhance the development of new workflow management software. Continuing previous research on service composition for interoperability, namely on DynamicFlow [8, 9], we focus on the background strategies that enable the streamlined execution of workflows with the development of the EU-ADR Web Platform.

Bioinformatics data integration environments were explored in this thesis with the development of WAVe [10, 11], an integrative platform for human variome information that uses innovative service composition strategies to collect and enrich data.

With this exploratory research work, a clear need for a different kind of strategies arisen. To build future-proof software we need to facilitate the use of strategies that will become common in the next years, from rapid application development technologies to the semantic web paradigm. Hence, we started the development of COEUS, a next-generation rapid application development framework [12-14]. With a prominent semantic web nature, this framework features a unique package with the tools to enable the integration of a wealth of information using type-specific data connectors and foster interoperability with external systems through a comprehensive API collection. The legacy Diseasecard portal was replaced with a new COEUS instance, validating and assessing COEUS role in a real-world scenario.

The aforementioned solutions were conceived after an evaluation of the challenges innate to the bioinformatics and computational biology fields as well as of the state-of-the-art software engineering technologies in place to engineer modern software architectures. As show in Figure 1-1, and detailed in the following section, this thesis is organized to facilitate the reading and comprehension of the problems at hand and of the devised solutions.

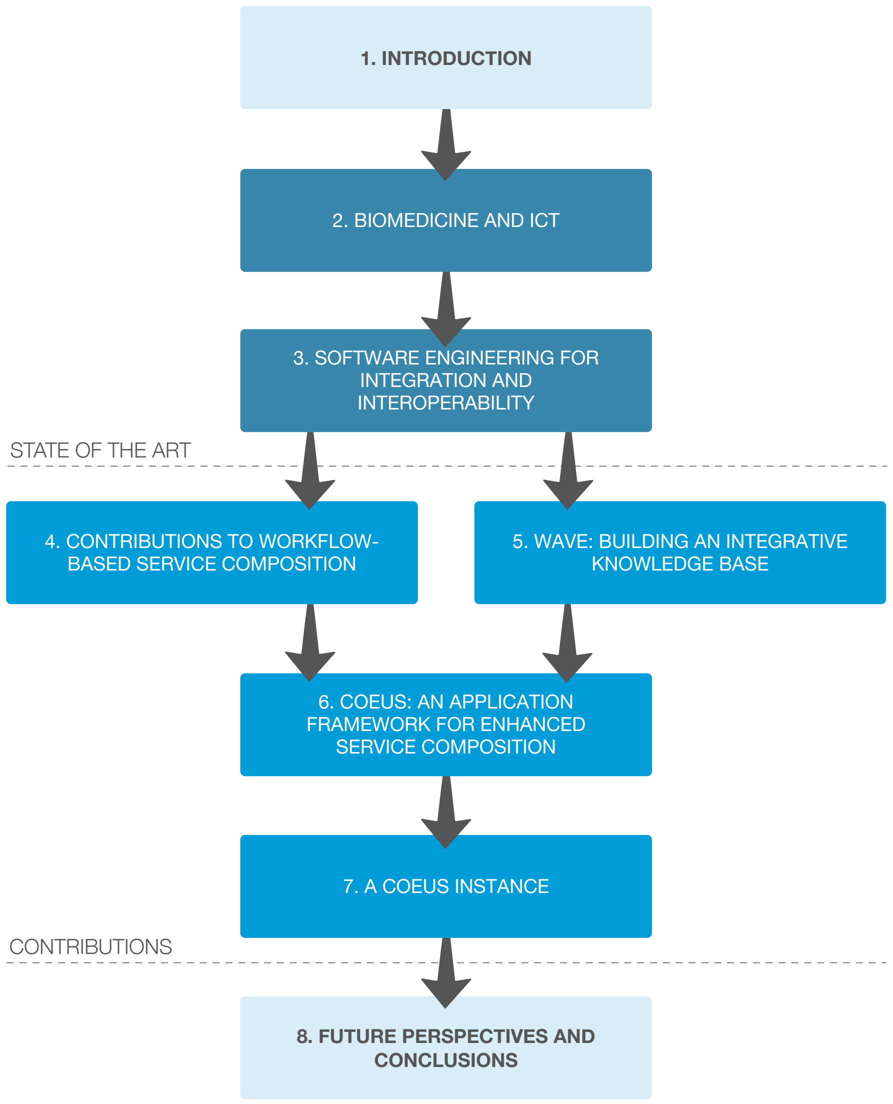

1. INTRODUCTION
2. BIOMEDICINE AND ICT
3. SOFTWARE ENGINEERING FOR INTEGRATION AND INTEROPERABILITY
STATE OF THE ART
4. CONTRIBUTIONS TO WORKFLOW-BASED SERVICE COMPOSITION
5. WAVE: BUILDING AN INTEGRATIVE KNOWLEDGE BASE
6. COEUS: AN APPLICATION FRAMEWORK FOR ENHANCED SERVICE COMPOSITION
7. A COEUS INSTANCE
CONTRIBUTIONS
8. FUTURE PERSPECTIVES AND CONCLUSIONS

Figure 1-1. Thesis organization. Chapters 2 and 3 compose the state-of-the-art evaluation, introducing the bioinformatics field and its problems, and analysing, in depth, the vast set of computational technologies available to tackle bioinformatics challenges. Chapters 4 to 7 form the overarching contributions of this thesis, highlighting innovative research and software products. Chapter 8 concludes this thesis.

1.3 Organization

The overall structure of the thesis takes the form of eight chapters. In addition to this introductory section, the thesis is organized as follows:

- **Chapter 2** - Biomedicine and ICT**.** This chapter contextualizes the thesis research work and provides clear background information on the problem at hand. It includes a brief walkthrough of the bioinformatics software landscape, identifying the most important data sources, services and applications. This overview exposes the most important challenges being faced in the biomedical software research field. This chapter is targeted at readers unfamiliar with the bioinformatics domain evolution over the last decades.
- **Chapter 3** - Software Engineering for Integration and Interoperability**.** This chapter is composed of an in-depth survey of mainstream computer science technologies that support modern service composition strategies. Starting with an introduction of service composition requirements and challenges, we move on to cover the best strategies for data integration and software interoperability and finish discussing the Semantic Web paradigm. This technological evaluation underpins the vital role that this broad set of technologies will play in responding to biomedical innovation demands. This chapter is directed to readers less experienced with the computer science background sustaining bioinformatics and computational biology innovation.
- **Chapter 4** - Contributions to Workflow-based Service Composition**.** This section details the exploration of workflow-based approaches towards the improvement of service composition in bioinformatics. Introduced contributions were performed within the context of the European EU-ADR Project and were part of the development of an advanced pharmacovigilance tool, the EU-ADR Web Platform. The chapter finishes with a discussion surrounding the use of workflows in bioinformatics environments.
- **Chapter 5** - WAVe: Building an Integrative Knowledge Base**.** In this section, one of the key outcomes from this thesis' research work is introduced: the Web Analysis of the Variome portal. Set within the human variome research and the European GEN2PHEN Project context, this discussion is centred on the produced technological advances that make WAVe an unique platform, from the innovative integration architecture to the exposed APIs. The final summary highlights how the

contributions from chapters 4 and 5 effectively bridge the gap between data and services within the life sciences domain.

- **Chapter 6** - COEUS: An Application Framework for Enhanced Service Composition**.** In this chapter another key contribution of this research work is reported, the COEUS framework. After a brief evaluation of existing tools for rapid application development in bioinformatics, we introduce the reasoning and developments underlying the creation of a new Semantic Web application framework. The COEUS platform is discussed in detail, covering all aspects surrounding the ability to create distributed knowledge networks.
- **Chapter 7** - A COEUS Instance**.** This section covers the deployment of the first COEUS instance. The reengineered Diseasecard is presented and its construction discussed as a sample for modern dynamic bioinformatics software, validating COEUS as a true enabler of rich knowledge ecosystems.
- **Chapter 8** - Future Perspectives and Conclusions**.** The final chapter of this thesis emphasizes the discussion surrounding future developments regarding the various developed strategies and exposes the conclusions of the conducted research work.

2. BIOMEDICINE AND ICT

"We said that once we had finished sequencing the genome we would make it available to the scientific community for free, ... And we will be doing that on Monday morning at 10am."
- J. Craig Venter

Bioinformatics is emerging as one of the fastest growing scientific areas of computer science. This expansion is fostered by the computational requirements leveraged by unprecedented advances in life sciences hardware and software.

This revolution kicked-off with the Human Genome Project (HGP) whose efforts resulted in the successful decode of the human genetic code [15]. HGP history starts in the middle of the 20th century with the involvement of the USA Department of Energy (DOE) in the first studies to analyse nuclear radiation effect in human beings. However, it took about 30 years, circa 1986, to propel and initiate the Human Genome Project. The ultimate project goals were as bold, audacious and visionary as the NASA Apollo program: to decode the "Book of Life" in its entirety. Moreover, this knowledge would be the basis of a new generation of tools that can identify and analyse a single character change in the sentences that compose our genetic sequence. Although HGP was an ambitious project, results appeared sooner than expected. This was the outcome of a deep collaboration with computer scientists that leveraged the deployment of novel software and hardware tools, aiding biologists' sequence decoding tasks. This joint effort between two large research areas, life and computer sciences, gave birth to a new discipline denominated bioinformatics.

The Human Genome Project brought about a variety of benefits in several fields. Remarkable discoveries in sequence decoding fostered DNA forensics, genetic expression studies, drug advances, and improved several other fields like molecular medicine, energy and environment, risk assessment or evolution studies. At the positively premature ending of the Human Genome Project, the availability of the human genome and other genome sequences have revolutionized all biomedical research fields [16].

Several projects started riding along HGP's success, using scientific discoveries and technological advances from HGP in miscellaneous scenarios to obtain new relevant information. On the one hand we have smaller projects, which are focused on specific genomic researches [17, 18]. On the other hand, we have larger projects that span through several institutions and cross various physical borders [19-21].

These projects originated a change on how everyone, from biologists to clinicians, assess and use computational tools. Information and communication technologies' role is nowadays vital in any biomedicine project's success. This happens to an extent where bioinformatics demands are constantly ahead of what computational technology has to offer, driving hardware and software innovation to new levels.

The ever-growing bioinformatics research brought about an infinite number of resources: databases, applications, services and protocols. Whilst this interest is essential to keep the research field dynamic and alive, the consequences of having too many systems with competing models, formats and technologies resulted in an incredibly fragmented software landscape. Alas, modern bioinformatics has to deal with common heterogeneity issues resulting from an anarchical ecosystem. Considering simple services used daily by bioinformaticians, such as PubMed literature search or BLAST sequence alignments, they have a similar technological backbone and probably adopt identical design principles and architectures. However, the user and data interfaces are entirely different in terms of output format, style and content. Whilst this variety is irrelevant for each specific field requirement set, the answers to most biological questions imply that the users browse, access and filter a myriad of distinct services until they have tracked down all the information they need. Thinking about large-scale projects where the proliferation of data is a defining feature, it is imperative to automate the acquisition and interoperation of data from both hardware devices and software systems.

Fortunately, important stakeholders are aware of these issues. Hence, current emphasis is given to converging synergies to produce much better outcomes. This is an area where the whole is more than the sum of its constituent parts, and this clear turn of events requires advanced computer science skills to revolutionize the way science is made, integrated and disseminated.

2.1 The "omics" Revolution

Genomic medicine evolution yielded great advances that reshaped how we generate, explore, evaluate and understand biomedical data. Likewise, the technological landscape was also reshaped with the breakthroughs of the last 20 years, originating an increased awareness of the complex composition of our surrounding environment. The improvement of analytical technologies and the availability of greater computational power were of extreme importance for molecular biology dynamics, resulting in an increased mechanistic understanding of ourselves. This lead to the development of various highly detailed fields, each approaching this data wealth from a different standpoint: genomics, proteomics, variomics, metabolomics, transcriptomics or pharmacogenomics, among others. Moreover, instead of viewing datasets independently, analysing genes, enzymes or proteins with a reductionist strategy, systems biology aims to exploit knowledge over different levels of molecular biology at once, assessing data as a whole.

As the "omics" revolution unfolds, with the rapid accumulation of data from a variety of distinct software and hardware bioinformatics tools, the simple standalone collection of these data does not suffice to understand the complete and dynamic system of life encrypted in our genetic material. Hence, this "omics" revolution demands a parallel technological paradigm revolution, promoting the evolution of independent closed legacy systems to futuristic open science integration and interoperability standards.

2.1.1 From the Genotype to the Phenotype

The genetic library that assembles life encrypts the heritability of gene-based disorders and defines our genotype. The complex relationship between our genetic features and environmental agents originates our phenotype. Despite the natural variability verified in human individuals, regarding weight, height, eye or hair colour, the human genome has a large common base [22]. In fact, most recent studies indicate that only 0.4% of our genetic sequence changes in 1% of the population [23]. Therefore, identifying and understanding individual or structural changes in our genotype is crucial to better explore the causes and consequences of the changes in our phenotype. Furthermore, the sequencing cost per genome is reducing drastically, faster than Moore's law, as shown in Figure 2-1, enabling access to a wealth of data like never before.

The collection of genetic mutations in the human genome is being researched in the Human Variome Project [19]. Similarly, the International HapMap Project looks to build the human haplotype map, describing human variation patterns [24, 25]. The 1000 Genomes Project adopts a distinct approach, focusing on the statistical analysis of multiple human genomes looking for correlations between common genotypes and phenotypes [26]. These projects, among others, influence decision-making in these research areas and provide the funding and opportunity to obtain further knowledge about us.

Nowadays the focus is divided in two parallel research lines, each with a distinct scope. On the one hand there is a broader perspective, directed to genome-wide association studies, analysing the genetic basis of complex traits like disease susceptibility and drug response through large statistical correlation studies. On the other hand there is a narrower approach, focusing on genetic mutations and their causes and effects on the human organism. Genome-wide association studies provide valuable insights over the genetic basis of some diseases, but lack the detail required for explaining disease heritability and propagation. Genetic variation studies generate large collections of granular data that in spite of being extremely precise, are missing richer connections for an overarching view.

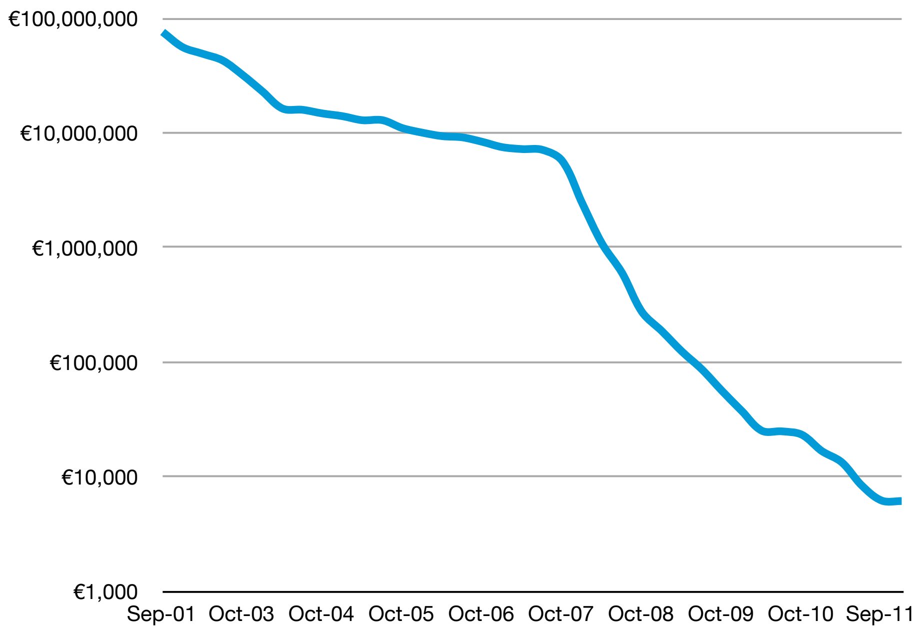


Figure 2-1. Sequencing cost per genome, the cost of sequencing a human-sized genome (logarithmic scale), from September 2001 (€76,210,457.6) to September 2011 (€6,194.4)[1].

Studying and understanding gene functions are essential steps to imply genes in human diseases. The Online Mendelian Inheritance in Man (OMIM) catalogue lists over 2000 diseases with single-gene or *Mendelian* disorders (diseases with simple familial inheritance patterns) [27]. Cystic fibrosis was one of the first *Mendelian* disorders to be identified and is caused by mutations in the CFTR gene on chromosome 7 [28, 29]. This particular rare disease affects approximately one person in 3,000. To figure out the roles of all players involved in life, from genotypes to phenotypes, including genes, proteins, drugs, enzymes,

[1] http://www.genome.gov/sequencingcosts/

pathways, and their interactions, is a demanding interdisciplinary challenge crossing the frontiers of life and computer sciences. Despite this, apprehending the meaning of the interplay between genotypes and phenotypes is vital to enable a more individualized healthcare.

2.1.2 Individualized Healthcare

Uncovering the genotype-to-phenotype intertwined relationship has the ultimate goal of improving individualized healthcare. Humans are much more genetically similar than different and, despite the population-based distinctions on phenotypes and diseases, only 5 to 10% of total human genetic variance occurs between populations and ethnic groups [30]. Nevertheless, most of worldwide population is being assaulted by similar conditions: cancer, obesity, diabetes or heart diseases are the main causes of death in first-world nations.

Individualized healthcare, through personalized medicine and custom drug treatments, is in a pivotal position for improving global health. This demands a strategy update, moving from traditional palliation care to directed cures. New therapeutic methods are targeted to specific disease processes, which further reduce the chances that patients will suffer from adverse drug events. By aiming at subgroups of patients with common genetic ancestry, we will be able to deliver the "right treatment to the right patient at the right time, every time" [4].

It is clear that this individualized healthcare evolution requires a parallel information and communication technologies transformation. Biomedical software and hardware technologies evolution culminates in a new transparent layer connecting the clinician with genetics information. To foster research in these areas, the European Union is promoting large funding initiatives, targeting vertical and horizontal integration and interoperability of data. We need to connect data from digital health records distributed through multiple region, nation or continent wide repositories. Furthermore, we need to connect data coming from genetic diagnostic labs to these digital repositories and deliver this rich knowledge to pharmaceutical researchers and clinical practitioners.

2.2 Connecting Life Sciences Data

To fully extract knowledge from the immense amount of data coming from omics research we need to foster the development of novel strategies and technologies to connect existing data and, above all, make these data available for future connections.

Life sciences innate complexity and heterogeneity require the most advanced computer science expertise to tackle the challenges associated with the harmonization, integration, interoperability and correct accreditation of data. This domain is extremely rich in applications, standards for services and data sources. The downside of this richness is its consequent fragmentation and enormous entropy. Randomly picking any life sciences research domain we can easily find various databases, services and applications, each with its own internal structures and data integration and exploration strategies.

Next, we detail our exploration of computational resources related to the multiple omics fields that drove this doctorate work and assess the challenges and demands arising for modern bioinformatics.

2.2.1 The Landscape of Information Access in Biology

The widespread availability of bioinformatics tools lead to an exponential increase in the amount of data available for researchers. Alas, the data-growing curve is steeper than the effective knowledge-growing curve. Being fairly easy to create new applications and services from scratch, anyone can launch new systems without taking in account any existing platform.

For all stakeholders involved in life sciences field, from wet-lab researchers to EU policy-makers, the amount of available information is overwhelming. More information also results in more applications, more platforms and more services. Nowadays, the information access landscape is a fragmented view, sinking in its own entropy.

The Nucleic Acids Research (NAR) journal keeps a collection of the most relevant biosciences databases, updated yearly. Figure 2-2 shows a graph showing the growth of this collection and the number of new databases. The 2012 edition adds 92 new databases to the 2011 list [31]. A sample overview of these lists' evolution over the last few years reveals an increasing complexity and specialization. Genetic variation datasets and single organism databases are growing where the number of overarching resources is steadier. This idea is a deciding factor to the new bioinformatics software path. Where niche fields were seldom targeted in the past, they are gaining relevance with area-centric databases and scientific curation.

This leverages a deeper background problem for connecting life sciences data. Fragmentation is increasing and these new focused systems are often built disregarding any integration or interoperability strategy. On the one hand, they do not integrate existing models or datasets, adding further entropy to the ecosystem. On the other hand, data are locked due to the lack of interoperability interfaces, blocking its use in other systems.

In summary, the access to and evolution of computer science technologies is a double-edged sword. On the upside, improved availability means that data are more easily at researchers' fingertips. On the downside, this also means deeper fragmentation. This leads to the branching of existing systems to uncoordinated areas, further augmenting the bioinformatics software landscape granularity.

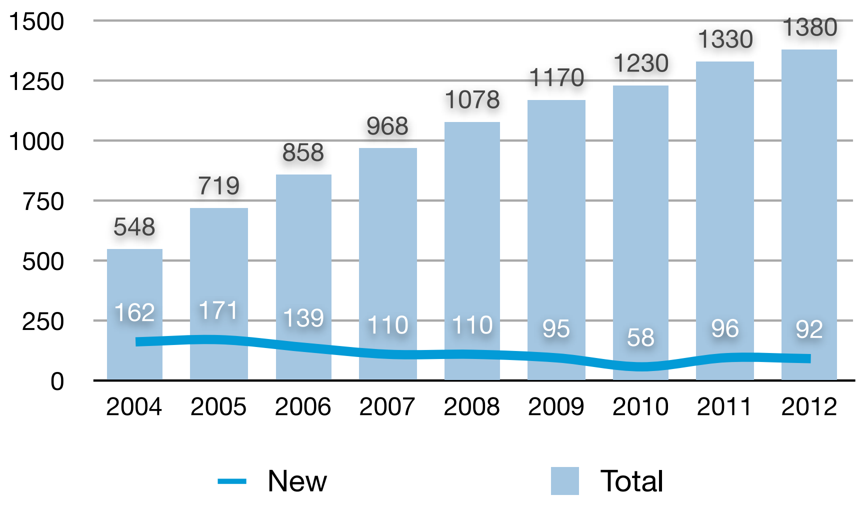


Figure 2-2. NAR database list evolution, from 2004 to 2012, regarding the total number of databases (bars) and the number of new databases (line).

2.2.2 Data Sources

Biological databases play a central role in bioinformatics. They offer scientists the opportunity to access a wide variety of biologically relevant knowledge, from reference sequences to marketed drugs. In most cases, databases offer their data through web services or flat files that can be easily accessed or parsed. However, these databases do not follow a single model or notation, and, therefore, the same biological concept may be represented in several distinct models and with various identifiers. The task of establishing relationships from one data type to other is often quite complex due to the multitude of existing data types, structures and domains.

Three large international players control the data sources landscape in bioinformatics in its majority: the United States of America National Centre for Biotechnology Information (NCBI), the partnership between the European Molecular Biology Laboratory and the European Bioinformatics Institute (EMBL-EBI), and the DNA Data Bank of Japan (DDBJ).

Notwithstanding the vital role played by the thousands of other data sources, registered in the aforementioned NAR registry, these three entities aggregate the funding and collaborations that enable the development of systems spanning the entire life sciences spectrum. Whilst the boundaries for each of these large data repositories are blurred, we can organize existing data sources according to their main controlling entity.

NCBI, associated with the National Library of Medicine in the USA, is a resource for molecular biology information organized in various categories each containing several databases. From the extensive NCBI database list we can highlight some major databases:

- dbSNP stores information about Single Nucleotide Polymorphisms (SNP), particular changes in our genetic sequence that are relevant for the detection of anomalies in our genes [32].
- The Mendelian Inheritance in Man (MIM) is a library of known diseases that are mainly caused by genetic disorders. NCBI was initially responsible for the Online MIM [27], which is now under John Hopkins University supervision.
- Medical Subject Headings (MeSH) is a thesaurus for medical terms, aggregating human health information in a tree-based ontology [33].
- Medical Literature Analysis and Retrieval System (Medline®) is a huge bibliographic database of published material referred to life sciences and biomedicine that can be accessed through PubMed, an online search engine.
- GenBank is an open sequence database that contains information from laboratories throughout the world and regarding a huge number of distinct species [34].
- The Entrez Global Query Cross-Database Search System (Entrez) offers online access to a multitude of NCBI databases through a single user interface [35]. Entrez is also a remarkable project on online resource integration, proving normalized data formats and coherency across databases and services.

At an European level, coordinated efforts between EMBL, EBI and the Swiss Institute of Bioinformatics (SIB) are the frontline of wide scale repositories, and have already left their footprint in the bioinformatics community. From these, the following data sources must be highlighted:

- UniProt is a universal protein resource, including a huge database of curated protein functional information [36]. In a smaller scale, InterPro is a competitor focused on proteins and the proteome [37].

- ExPASy is a proteomics resource portal, providing an entry point to a vast collection of SIB resources focused on protein knowledge [38].
- PROSITE is a protein domain data source, focusing on annotating functional products and features [39].
- ArrayExpress archives public functional genomics data in two databases [40]: 1) Experiments Archive that stores results from conducted experiments submitted from the entire world. 2) The Gene Expression Atlas is a curated and re-annotated subset of the Experiments Archive that is directed to gene expression studies.
- Ensembl is a genome database that contains information from a large number of species and is accessible through a large number of web services [41].
- The European Genome-phenome Archive[2] (EGA) and the 1000 Genomes Project collect datasets with complete sequence information from multiple individuals [26].

DDBJ cooperates with NCBI and EMBL-EBI to provide data replicas to the Asian market and to develop new tools and data sources. The most relevant outcome is the Kyoto Encyclopaedia of Genes and Genomes (KEGG), collecting genomic information relevant to metabolic pathways and organism behaviours [42]. KEGG is composed of five main databases, each with a distinct focus: Pathways, Atlas, Genes, Ligand and BRITE. With all these databases, KEGG's goal is to obtain a digital representation of the biological system [43].

Besides the data sources associated with these three corporations, there are many more high quality databases, often targeting niche-focused fields, from gene ontologies to phenotype information:

- Gene Ontology is the most widely accepted ontology, aiming to unify the representation of gene-related terms across all species [44]. This is only possible by providing access to an annotated and very rich controlled vocabulary [45].
- PhenoGO is a Gene Ontology centric database that intends to support high throughput mining of phenotypic and experimental data [46, 47].
- PhenomicDB is a database for comparative genomics regarding various species and genotype-to-phenotype association. Information is obtained from several public databases and merged in a single database schema improving database access performance and making several other features possible [48, 49].

[2] https://www.ebi.ac.uk/ega/

- PharmGKB is a data source aiming to bridge the gap between pharmacogenomics and bioinformatics through the establishment of curated association between genes and drugs [50].
- GWASCentral collects and summarizes genome wide association studies, making them more easily available for researchers [51].

Semantic Web awareness has also been increasing within the life sciences community [52-54]. Hence, many existing knowledge bases are emerging, adopting new semantic web paradigms and looking for their space in a very competitive market:

- Bio2RDF is the most relevant development [55]. Bio2RDF falls on the design of an enhanced strategy for semantic data warehousing, complete with a complex Extract-Transform-Load pipeline that enables the collection of millions of records from the most relevant life sciences databases.
- Bioportal is a portal for the integration of ontologies in the life sciences domain, which has grown to become the de facto location for biomedical ontology exploration [56, 57].
- The LinkedData initiative is also strongly present in the life sciences field, where multiple computational biology datasets are already published according to the proposed guidelines [58-60].

On a broader field, DBPedia is a notable development, providing a semantic version of Wikipedia data available through a public SPARQL endpoint and with rich internal crossed relationships [61].

2.2.3 Services and Providers

Data management in life sciences offers constant challenges to software engineers. Offering these data to end-users and researchers worldwide is an even bigger challenge. Web applications tend to be complex and cluttered with data resulting in non-usable interfaces and fragile workspaces. The possibility to offer data as a service is a valuable option that is being used more often. The greatest benefit of these remote services is that they allow static or real-time dynamic programmatic service composition. That is, developers can merge several distributed services in a single centralized application.

Nowadays, most of the previously mentioned data sources provide access to their internal knowledge base through a rich set of services. On top of these, there are many other relevant web service standards and providers. Next we enclose a small revision of these tools, from service protocols to registries.

- The Distributed Annotation System (DAS) specifies a protocol for requesting and returning annotation data for genomic regions that has expanded to several life sciences areas, not only sequence annotation [62]. The main idea behind DAS is that distributed resources can be integrated in various environments without being aware of other intervenient. That is, resources can be replicated and integrated in several distinct systems, not only in a single static combination of resources.
- BioMart consists of a generic framework for biological data storage and retrieval using a range of queries that allow users to group and refine data based upon many different criteria [63, 64]. Its main intention is to improve data mining tasks and it can be downloaded, installed and customized easily.
- The European Molecular Biology Open Software Suite (EMBOSS) is a software analysis package that unifies a collection of tools related to molecular biology and includes external service access [65-67]. Applications are catalogued in about 30 groups ranging several areas and operations related to the life sciences.
- Soaplab was developed at the EBI and is another set of web services that provide remote programmatic access to several applications [68]. Included in the framework are a dynamic web service generator and powerful command-line programs, such as support for EMBOSS software.
- BioMOBY is a web-service interoperability initiative that envisages the integration of web-based bioinformatics resources supported by the annotation of services and tools with term from well-known ontologies [69, 70]. The BioMOBY protocol stack defines every layer in the protocol from the ontology to the service discovery properties.
- The Web API for Biology (WABI) is an extensive set of SOAP and REST web life sciences APIs, focused on data processing and conversion between multiple formats [71, 72]. WABI defines mainly a set of rules and good-practices that should be followed when the outcome of a research project is a set of web services.
- Biocatalogue is an attempt to facilitate the discovery of existing web services [73]. In this library, users can register, discover and annotate bioinformatics web services. Taking advantage of this community-based approach, Biocatalogue complements its internal service monitoring system with crowdsources curation, improving the assessment of existing web services.

2.2.4 Applications & Frameworks

Goble conveyed a “state of the nation” in bioinformatics study and her main conclusions were that there is still a long path to traverse, specially concerning integration and interoperability efficiency [74]. Nonetheless, there were remarkable developments in the last few years. These developments include novelties in data and services integration, semantic web developments and the implementation of mashups/workflows in bioinformatics. Moreover, as stated by Stein, integration strategies are vital to the creation of a large bioinformatics ecosystem [75, 76].

The diversity of strategies results in a myriad of applications and frameworks that tackle similar challenges. We can group these approaches under three concepts, according to the basic functionality of each tool: integrative tools and applications; frameworks and development libraries; and workflow managers.

First, the set of available software focusing on integration heterogeneous bioinformatics components is nearly immeasurable. Everyday new applications appear, targeted to distinct fields, end-users or operating environments. The following listing highlights some of the existing applications with particular focus on some in-house solutions:

- GeneBrowser adopts a hybrid data integration approach, offering a web application focused on gene expression studies, which integrates data from several external databases as well as internal data [77, 78]. Collected data are stored in an in-house warehouse, the Genomic Name Server (GeNS) [79].
- Biozon is a data warehouse implementation similar to GeNS, holding data from various large online resources like UniProt or KEGG and organized around a hierarchical ontology [80]. Biozon clever internal organization (graph model, document and relation hierarchy) confers a high degree of versatility to the system, allowing a correct classification of both the global structure of interrelated data and the nature of each data entity.
- Reactome is a generic database of biology, mostly human biology, describing in detail operations that occur at a molecular level [81].
- BioDASH is a semantic web initiative envisaging the creation of a platform that enables an association, similar to the one that exists in real world laboratories, between diseases, drugs and compounds in terms of molecular biology and pathway analysis [82].

The development of bioinformatics applications has been enhanced greatly in the last decade. This was possible due to the appearance of multiple libraries and frameworks, targeting miscellaneous development environments, and enabling the creation of new software at a much faster rate. From the broad set of bioinformatics packages, we must highlight the following:

- BioJava is an open-source framework that eases the development of bioinformatics applications by providing packages facilitating access to widely used databases and services [83]. Similar libraries are also available for other programming languages such as Ruby [84], Perl [85] or Python [86].
- Bioconductor is another open-source and open development software package providing tools for the analysis and comprehension of genomic data [87]. The software package is constantly evolving and can be downloaded and installed locally. The tools that compose the package are made available from several service providers, generally in R language.
- Molgenis is a rapid application development framework enhancing the creation of new bioinformatics web information systems with only a couple configuration files [88]. Molgenis is introduced in detail in section 6.1.1.

For service composition, mashups or workflows are among the hottest trends in bioinformatics application development. Service composition, which encompasses service orchestration and choreography, is already possible in various scenarios:

- myGRID is a multi-institutional and multi-disciplinary consortium that intends to promote e-Science initiatives and projects [89]. More recently, GRID is giving place to cloud-computing strategies, a field that is still lacking interest in the bioinformatics community, though it will gain relevance in a near future [90, 91].
- Bio-jETI uses the Java Electronic Tool Integration (jETI) platform, which allows the combination of features from several tools in an interface that is intuitive and easy to new users [92]. jETI enables the integration of heterogeneous services from different providers or even from distinct application domains.
- BioWMS is an attempt to create a Taverna-like web based workflow enactor. The set of features is not as complete as Taverna and the availability is very limited (unlike Taverna, which is available freely for the major operating systems) [93].
- The Workflow Enactment Portal for Bioinformatics (BioWEP) consists of a simple web-based application that is able to execute workflows created in Taverna or in

BioWMS [94, 95]. Currently, it does not support workflow creation and the available workflow list is quite restricted.

- The Bioinformatics Workflow Builder Interface (BioWBI) is another web-based workflow creator that connects to a Workflow Execution Engine (WEE) through web-services to offer complete web-based workflow enactment [96].
- BioFlow [97, 98] is a new generative, declarative query language that permits the exploitation of services, databases or ontologies for data integration.
- Taverna is the best state-of-the-art application regarding workflow enactment [99]. It is a desktop application that enables the creation of complex workflows allowing access to files and complex data manipulation. Additionally, Taverna also configures, automatically, the access to BioMOBY, Soaplab, KEGG and other services. Along with these predefined services, users can also dynamically add any web service through its WSDL configuration.
- Galaxy is Taverna's most competitive alternative [100, 101]. This web-based platform provides an online workspace for managing and executing workflows, enabling the tracking of provenance data and the reproducibility of bioinformatics research.
- Tavaxy attempts to combine the best features from Taverna and Galaxy in a standalone bioinformatics integration and interoperability platform [102].

2.2.5 Challenges for Modern Biomedical Software

Many problems and requirements arise with life sciences research and technological evolution can only do so much. The bioinformatics advances we are witnessing require a strategy shift. We must tackle the overwhelming growth of data sources with novel strategies to synthesize, harmonize and (re) connect data.

With an ever-growing data supply, researchers are given the task of assessing what is relevant and not. Therefore, summarizing the huge datasets traditionally available is a first step towards a better bioinformatics ecosystem. This also fosters the need for data curation. Whether through social crowdsourcing strategies or direct data analysis and validation, the wealth of life sciences data must be carefully curated to extract knowledge as precise and accurate as possible. Moreover, data selection is also hindered by the well-known researcher accreditation fear. The case with the majority of large-scale data sources is that original data authors are lost along the way. Content ownership and authorship is forgotten, removing credit from where it is due. Novel approaches such as microattribution and nano-publications try to adapt the publication of data to similar strategies as the ones with the publication of research literature, highlighting authors' accreditation [103].

The harmonization of data relates directly to the "reuse instead of rewrite" principle. While innovation is achieved with new outstanding ideas, developers should first study what has been done previously in a specific domain. This incorrect assessment results in the key fragmentation issue: heterogeneity. Heterogeneity of data formats. Heterogeneity of data structures and models. Heterogeneity of data access methods.

We can draw a linear path to wisdom. We acquire data then we translate it to information, which allows us to gain new knowledge. However, to make the most efficient transition from data to wisdom we need to fully explore the connections amongst independent data units. Only by interlinking all bits of data we can obtain a rich, holistic, overarching view of everything we have collected and, ultimately, to make sense of it.

With the increasingly growing amount of data and increasingly growing amount of ways to deal with it, we are faced with the challenge of evaluating where the most relevant data are stored, who created it, how we can connect it with our research, and how we can make our results available to others in the future. Answering these questions will provide us the path to improving our wisdom regarding the life sciences fields, and to accomplish this we must seek the integration of our resources and foster interoperability amongst them.

Integration

Integrated data views are essential for a better understanding of existing datasets. In the majority of scenarios, the exploration of scientific results involves establishing connections between data from diverse domains. In his workspace, a clinician needs direct access to patients' records, disease data and drug details. In the genetics wet lab, researchers require hardware sequencing tools with connections to data pipelines and sequence aligners or ontology browsers. In the office computer, researchers need the best set of tools to explore wet lab data, select publications or navigate through a myriad of related resources.

In practical terms, analysing items individually results in poor quality views, which ultimately imply inaccurate and non-precise results. There is a demand for more comprehensive integrated data views, setting a wide number of resources at the users' fingertips. For example, the *in silico* research scenario requires BLAST tools, protein data from UniProt and PDB, literature views from PubMed and gene information from HGNC database. Despite being a simpler scenario than what occurs in a real lab, we already need access to 5 distinct resources, each with own independent data structures, exchange formats and access methods.

Whether we are providing virtually centralized integration or replicating data or features in their entirety, integration deals with the strategies for getting something from an external, remote or distributed environment into a centralized tool - Figure 2-3.

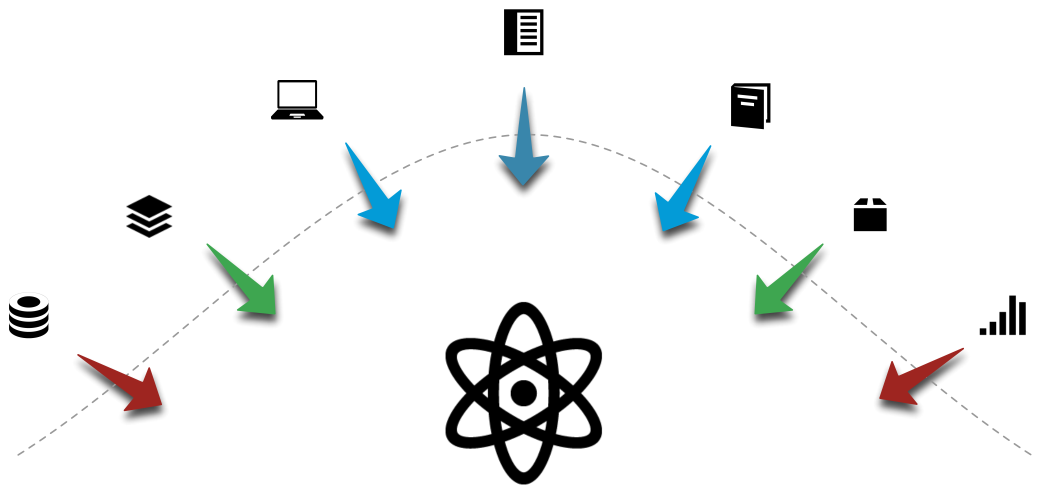


Figure 2-3. Integration of distributed and heterogeneous resources. The idea behind integration is to physically or logically move data from an external resource into a new virtually centralized location.

With the ever-increasing heterogeneity of the bioinformatics domain, creating these integrated data views is especially difficult. Furthermore, considering the exploration of niche scenarios, separating the wheat from the chaff is worse than finding a needle in a haystack. Therefore, the most advanced resource integration strategies adopt service composition approaches. Considering each distributed component as an autonomous entity that can be connected with multiple others, with whatever software artefact, enables simplifying the overall integration algorithm view.

Interoperability

A key feature for the integration of resources is inward interoperability. Being it a straightforward process on a forced one, we need to make our centralized integrated environment interoperate with some kind of external player. In an ideal scenario, if the resources being integrated already account for modern outward interoperability features, the tasking of integrating its features and/or data are facilitated.

In the previous integration scenario, the involved entities already provide a collection of web-based services enabling streamlined access to their data. Hence, it is fairly easy to develop a new static application to read and process RDF data from UniProt, XML data from

PDB and PubMed and CSV data from HGNC. In spite of being fairly easy to accomplish this, most data remains locked in independent closed data-silos or available through legacy methods.

For this matter, we need to make a sure bet on improving existing systems' interoperability features and on including the adequate interfaces in newly built software. This way, applications and services can be composed to form new dynamic software ecosystems, where data are easily exchanged from tool to tool - Figure 2-4.

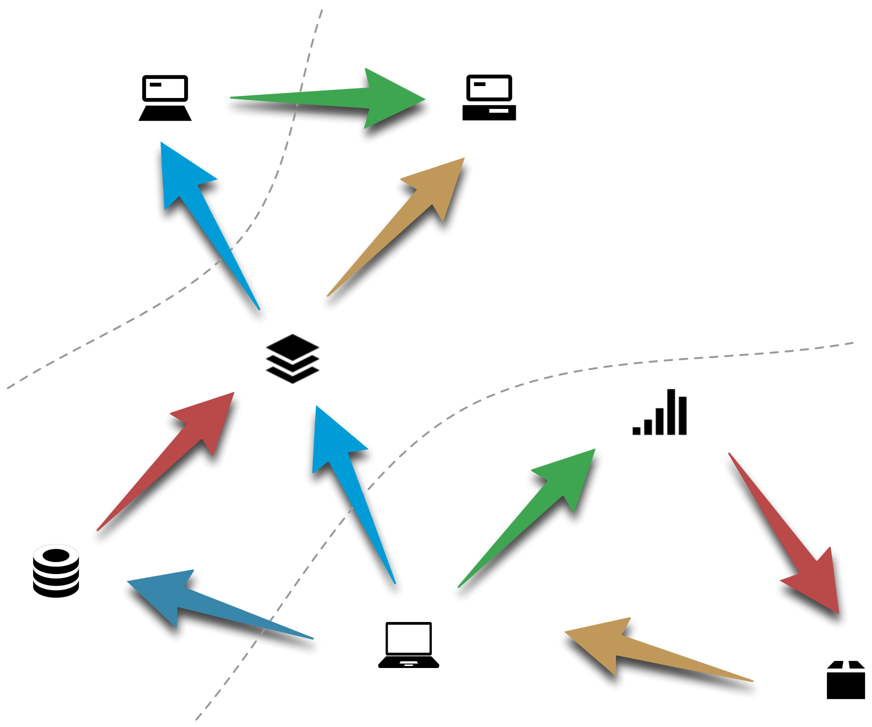


Figure 2-4. Interoperability amongst distinct resources. With autonomous software interoperability resources are able to dynamically exchange and accurately interpret data.

Developers and researchers are already endowed with advanced interoperability frameworks. Whether through resource-specific data access or standardized service-based methods, developers are able to access and publish information easily. Nevertheless, in an ideal scenario with all data promptly available, the heterogeneity issue strikes again. Even when the data models are compatible, developing software to interoperate with multiple services is not trivial. Furthermore, researchers have a new set of demands that cannot be satisfied with the previously mentioned static application example.

With more data and more services, it is of utmost importance to select the best alternatives for each field to enable the creation of dynamic interoperable software that not only allows the integration of heterogeneous remote data, but also promotes the future exploration of collected data through an advanced set of APIs.

2.3 Discussion

2.3.1 Enabling Bioinformatics

It is essential to grasp the true issue behind the bioinformatics software ecosystem to understand what needs to be done to overcome current challenges. Researchers demanded more data from their hardware and software platforms, resulting in an explosive data growth that has given more than they asked for. In a way, the problem of getting quality proof data is still present. Two decades ago it was cumbersome to get any data regarding any life sciences subject due to the lack of adequate databases and services. Nowadays, it is cumbersome to select the best resources from the overwhelming amount of data and services in most of cases providing similar content.

In an ironic turn of events, the tools researchers proposed to solve a general problem are now causing an entire new kind of challenges. Researchers demand new tools for synthesizing, connecting and harmonizing available data. With this, the fields of software integration and interoperability appear as the most viable solution for enhancing access to life sciences knowledge, in a role that resembles the one played by scripting languages and relational databases in the early days of bioinformatics.

Not only we need to improve how we collect, filter and select the best data in a given domain, we also need to make these acquired and enriched data available to other players through interoperable methods. To accomplish this we need to foster the development of a new generation of bioinformatics software. This requires us to devise new strategies and explore modern technologies that will enable the future of bioinformatics.

This new generation of bioinformatics tools will involve efforts from all stakeholders in the life sciences domain. Wet-lab researchers, clinicians, principal investigators and policy makers must be aware of where this uncontrolled evolution is going. While in the early 21st century scientist could rely on the quality of existing databases without questioning their content, the overwhelming heterogeneity issues are leading us to an undisputable quality loss.

To enable modern bioinformatics, life sciences researchers and enterprise stakeholders need to adopt new principles in their development process, rethinking how existing information can be reused, i.e. integrated, and how it can endure the force of time, i.e. made interoperable. For these matters, service composition strategies arise as the optimal solution to help the field of bioinformatics thrive in the future as it has in its recent past.

2.3.2 Standardizing Bioinformatics Services

Despite the last decade's bioinformatics evolution, the use of web services in the field remains rather primitive. On the one hand, web service use is pervasive and their actual execution has been largely facilitated on the last couple years. However, on the other hand, this growing amount of services also makes it cumbersome to select the fit service for a particular task. This results in an overwhelmingly chaotic services landscape.

With a growing number of services, it easy to assess the struggle for building applications that routinely used distributed web services. In this context, several notable initiatives to standardize bioinformatics service composition emerged. Services and protocols highlighted in this chapter were the initial attempts to produce a broad web service standard for bioinformatics. However, these technologies never gained enough traction to become the mainstream *de facto* service standard. Whilst the standards themselves are actually good, they sit atop the traditional service protocols, adding a new complexity layer. Furthermore, the use of bioinformatics service standards hindered their adoption from new developers. When implementing new solutions from scratch, the lack of consensus regarding the best practice propelled most developers to create their own protocols, further fragmenting the field.

With so many services from so many distinct providers to choose from, the bioinformatics developer community is faced with a daunting challenge in the form of choosing the best service for a given set of problems. This diversity arises in the form of standards, programming languages, independent packages and/or frameworks. This means that non-expert users have a though call to make when selecting an existing service standard to build upon. To tackle these challenges, new strategies started to look at original ways to enable service composition, namely through the adoption of workflow strategies served in high-end user interfaces.

2.3.3 Service Composition for Biomedical Applications

In this chapter we highlighted the multitude of solutions devised to overcome past bioinformatics and computational biology problems. However, a new set of demands arises from this technological evolution. Therefore, and to avoid this vicious circle in the future, we must adopt novel software integration and interoperability paradigms.

From a computer science perspective, the composition of web services is essential to enhance the development of state-of-the-art bioinformatics software. Software engineering experts must endow bioinformatics developers with tools and reusable assets to promote a new generation of biomedical applications. These will allow the deployment of software that will prevail over the multitude of drawbacks inherent to the need for synthesizing, connecting and harmonizing knowledge.

To this end, service-oriented architectures and novel data integration and software interoperability strategies must be evaluated, allowing the creation of guidelines for delivering enhanced software to development partners. It is our belief that the pursuit of better integration and interoperability platforms should drive bioinformatics development in upcoming years.

3. SOFTWARE ENGINEERING FOR INTEGRATION AND INTEROPERABILITY

"Perhaps, I may get well if you will let me study engineering."
- Nikola Tesla

In the middle of the 20th century the computational industry started to grow sustained by an evolution in hardware components. During almost 4 decades, until the birth of the modern PC, computers changed mostly in format and size. Only in the early 80s we witnessed a major leap in available software and user interactions, brought about by innovation from companies such as Apple, Microsoft or IBM and the advent of the Internet.

From there on, in the PC era, we can safely assume that strategic changes were promoted by a software revolution despite the undeniable key role hardware has played. Software engineering has evolved to better use available hardware resources and transformed the way the world sees computers. At any given moment, all major sectors are using some kind of computer software tool. Whether it is to trade stocks, manage production factories or analyse sequencing data, computer science is essential to keep the world moving forward.

Nowadays we are entering the post-PC era. Hardware advances have taken another leap, this time into our pockets. With the mobile device number already largely surpassing the number of desktop computers, software development paradigm changes are once again required to push available hardware to its limits, exploring all the new ways we can interact with machines and new ways machines can interact amongst themselves. Modern software development revolves around the concept of web services. In a broad sense, web services can be seen as any computational software feature available through a web-based interface.

Services are (should be) built to be composed, to play together in an ensemble of interactions amongst heterogeneous and distributed actors. Hence, the emergent relevance of service composition is vital. In a world replete with a growing number of data coming from all kinds of digital sensors, research labs, industry and entertainment products and, above all, ourselves, the idea of providing these data as a service that can be used to build new intelligent software ecosystems is deeply attractive.

To achieve this, we need to be able to include a wide variety of features and data in any modern software system so that users' demands can be fully satisfied. In a sense, users want to have everything at their fingertips, a couple touches or clicks away.

This need is further highlighted in the life sciences research domain. The great computational hardware and software leaps are occurring now in bioinformatics as new technologies are generating more and more data, which in turn results in more and more services, which ultimately results in an overwhelmingly heterogeneous landscape. Researchers are now crushed beneath the house they have built due to the growing entropy in their fields and increasing difficulty in getting the best data. From the life sciences standpoint, this is the perfect opportunity to introduce state-of-the-art technologies in the bioinformatics domain: modern service composition strategies are perfect to satisfy integration and interoperability demands, widely common in the biomedical domain.

This chapter covers in depth the technological requirements and state-of-the-art solutions regarding modern software engineering. Integration and interoperability strategies are discussed, leading the way to the introduction of Semantic Web technologies as the best-of-breed paradigm to employ when developing new systems.

3.1 Rethinking Software Engineering

Regarding software, new application development paradigms are being implemented with a more problem-oriented perspective. Instead of focusing on solving a single goal, modern software tries to tackle multiple challenges at once, providing new tools with incredibly wide feature sets. This demands a more integrative approach. We can no longer aim to build an entire data-rich and feature-rich ecosystem from scratch without considering what has been done before. We are not expected to build a social application without connections to Facebook or Twitter. Likewise, bioinformatics developers are not expected to build a new proteomics resource without integrating data from the UniProt knowledge base, for example.

Moreover, applications are no longer created solely for end users. There is a growing concern in providing tools that allow other developers to build upon the initial system. This implies that software interoperability features are essential for an application success and developers are now more prone to including them in their software.

This "reuse instead of rewrite" phenomenon is also observed in the application development process. More often we now see applications build on top of existing frameworks and APIs. Rapid application development strategies are in place to reduce the time-to-market for new applications, leveraging on common features or data that will undoubtedly be required in a given domain.

For computer scientists and, more specifically, software developers, this means that there is an entire new set of technologies and strategies to be explored. New applications are more connected with deeper integration features and broader interoperability tools.

3.1.1 Development Paradigms

We cannot assert that there are perfect strategies as the problems themselves change, especially in an area as dynamic as the life sciences. Whereas for reading data from a database a simple static shell script is enough to obtain the desired results, in complex data integration environments more robust and complex solutions are required.

Service composition for biomedical applications can assume many forms, ranging from the mentioned simple *ad hoc* approaches to advanced software engineering projects. Whether we are building static applications or a modern workflow manager we need first to take in account what exists in the area and what are the real demands to make a correct assessment of the best strategy to use.

Static Applications

The simplest approach to use service composition strategies for the integration of heterogeneous components is to implement the entire application workflow directly. These applications combine a collection of methods to integrate each resource. As a result, a static application is composed of a set of wrappers that encapsulate the access to distributed data resources. At first sight, these applications do not represent a valuable solution for the integration of resources. Nonetheless, this solution is widely used specially due to the simplicity of the development and the speed of deployment. With static applications, developers do not need to program dynamic or generic components. On the downside, static applications are not generic, flexible or robust. Anytime one wishes to add a new resource, developers must program the access to that particular service and add it to the application. Whilst this solution is feasible at a small level, when we are dealing with complex environments and a constantly evolving scenario it is not enough.

Dynamic Applications

The design and development of dynamic solutions requires a higher-level of computer science expertise and background on the research area to support the various iterations of the project execution. Dynamic applications are the expected evolution of static applications and are distinguished for being able to allow changes in its inputs and outputs as well as conveying distinct service combinations to reach a given goal [104, 105].

Dynamic access to external services or autonomous service composition requires the development of several focused, flexible and generic integrative middleware protocols [106]. Designing these protocols implies recurring to a multitude of distinct technologies and requires the adoption of advanced strategies to describe integrated resources and permit the interoperability with novel ones. Despite this new complexity layer, dynamic applications' generality makes them a more suitable solution for complex software environments.

Meta-applications & Workflows

Metadata are data about data. Applying the same premise to applications, we conceive the paradigm of meta-applications: applications working over applications. Meta-applications are state-of-the-art systems that connect distributed applications enabling interoperability among heterogeneous systems. Recent developments have also promoted a concept described as "software-as-a-service": any software can be provided as a remote service [107]. If software engineers follow this paradigm, any application could act as a service, easing the composition tasks.

The mashup term characterizes hybrid web applications: applications that mesh applications. Their purpose is to combine data gathered from multiple services to offer a wider centralized feature [108, 109]. Mashups allow easy and fast integration relying on remote APIs and open data sources and services [110]. Mashups and meta-applications share a common basic purpose: to offer a new level of knowledge that was not possible by accessing each service independently.

According to the Workflow Management Coalition [111], a workflow is a logical organization of a series of steps that automate business processes, in whole or part, and where data or tasks are exchanged from one element to another for action. Adapting this concept to software, a workflow is a particular implementation of a mashup that consists on an ordered information flow that triggers the execution of several activities to deliver an output or achieve a goal (Figure 3-1) [112-116]. A crucial workflow requirement is that the inputs of each activity must match with the precedent activity outputs to maintain consistency. Dealing with workflow execution operations requires the implementation of workflow management systems [117].

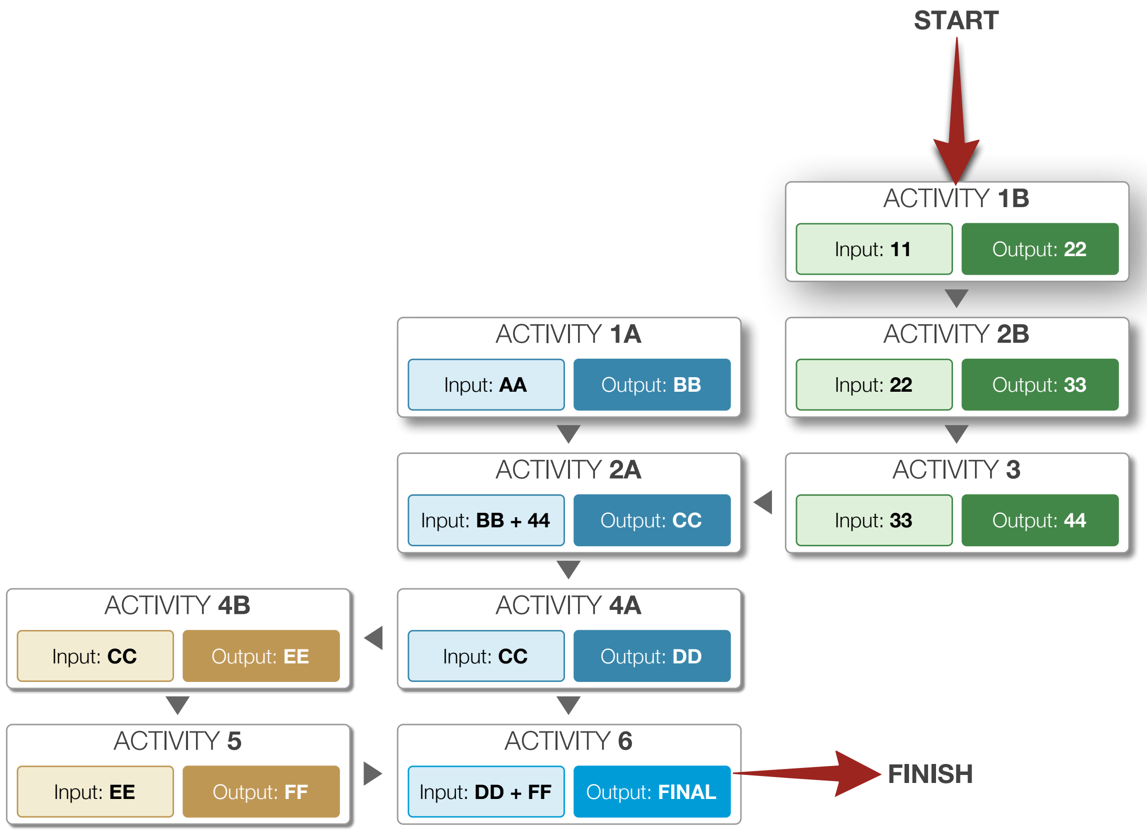


Figure 3-1. Workflow example. Internal activities (independent tasks) are executed at distinct providers and can operate within various domains.

A workflow management system allows designing, controlling and executing software that is driven by a computational representation of the workflow logic. Describing the workflow requires a complete description of its elements: task definition, interconnection structure, dependencies and relative order. The most common solution to store workflows is to use a description language, a configuration file or a database.

Existing workflow systems can support complex operations and deal with large amounts of data. Though, there are emerging requirements that must be handled by workflow management systems, such as provenance, event-driven activities, streaming data and collaboration between personnel in distinct parts of the globe. In research domains such as the life sciences, researchers prefer to design, execute and analyse the workflows in real-time.

3.1.2 Promoting Rapid Application Development

From an implementation perspective, Rapid Application Development strategies (RAD) are a software development method resembling high-level Agile, spiral, waterfall or SCRUM ideals [118, 119]. The grounding for RAD relies on its focus on quickly prototyping entire applications in opposition to having an intensive planning stage. This allows for a quicker time-to-market development process sustained by quickly developing an initial application version, shown to final clients, which is actively iterated until the final application is ready for prime time.

At first, RAD strategies where used to deploy initial application prototypes with only the application skeleton build. However, with the advances in software engineering, RAD strategies become more prominent, enabling the use and combination of multiple skeleton components in the final software system. Based on a set of architecture configuration files, customizable components (databases, user interfaces, services...) are automatically generated and combined with other reusable assets. This pushed RAD as a serious development methodology, spreading its adoption to every software area.

RAD strategies found wide use in web-based applications [120]. Programming environments such Ruby on Rails[3] and web development frameworks such as Zend[4], Symfony[5], Django[6] or Grails[7], among many others, provide developers with the tools to quickly create new complex web information systems. These packages include database abstraction layers, MVC support, wrapping APIs and easy web service creation facilities. Once familiarized with these frameworks, developers are endowed with the minimal set of tools to create their applications almost instantly.

The application of RAD strategies to user interface development is also witnessing an increased use. Frameworks such as Boostrap[8], endFoundation[9] or HTML5 Boilerplate[10],

[3] http://rubyonrails.org/

[4] http://framework.zend.com/

[5] http://www.symfony-project.org/

[6] https://www.djangoproject.com/

[7] http://grails.org/

[8] http://twitter.github.com/bootstrap/

[9] http://foundation.zurb.com/

[10] http://html5boilerplate.com/

include a diverse set of predefined user interaction components. This reduces the amount of work required to build new interfaces and enables the creation of new integrative toolsets based on customizable components in a LEGO-like way.

The entire software development process is changing in the post-PC era. Developers are now looking for solutions that deliver more features with the smaller cost, i.e. with less programming required. This is particularly evident in modern cross-platform application scenarios, where the server-side and client-side components are reused even when the final application targets distinct operating systems or working environments.

3.2 Understanding Service Composition

We have discussed the need for creating a more dynamic application ecosystem, fostered by all fronts demanding improved interactions, flexibility, robustness and performance in addition to the software fragmentation originating in a hardware market surrendered to mobile, tablets, TV and web alternatives to the traditional desktop box. The glue to maintain these rich architectures together is service composition. Service-oriented architectures allow accessing online resources to obtain integrative views and promote machine-to-machine interoperability through standardized data exchanges.

Service composition comes in two flavours [121]. On the one hand there is service choreography, where each participant web service controls its own agenda. On the other hand there is service orchestration, where a master node controls the coordinated execution of distributed services.

Choreography approaches require a higher level of intelligence from each web service. Services act autonomously based on a set of predefined rules to achieve the desired goal. This means that artificial intelligence measures must be in place to manage the interaction of multiple independent actors to yield an overall composition result. In a sense, service choreography operates like an artistic dance performance, with each element working together to deliver the best possible show.

Orchestration is a more easy and therefore common approach, requiring a controller node to define the sequence of tasks within a workflow. These logic actions are traditionally static and defined by developers. It is a centralized approach in contrast to choreography's distribution, resulting in the problematic of having a single point of failure. Orchestration, as the name says, is similar to an orchestra playing, where the maestro controls the arrangement and is able to predict or solve problems on demand.

3.2.1 Accessing Resources

Accessing online resources is no longer a challenging task for computer scientists. The set of tools available for every development environment is enough to allow the creation of advanced data access platforms. Online resource access services are responsible for encapsulating data or features and making them available to other systems. They allow access to databases, tools, file systems or any kind of external storage methods. The services encapsulation should be made using wrappers to enhance and ease integration and interoperability. With this in mind, it is important to take into account some generic concerns:

- **Performance** is a crucial concern especially due to the fact that end-users want fast and responsive applications regardless of the operation they are executing. Performance can be optimized by reducing the amount of data that is sent across the network or by minimizing query interdependence, thus reducing latency.
- **Usability** is also essential in any modern system. Expressiveness should be enough to allow application developers to pose almost any query to the system. Usability and expressiveness depend on metadata. Metadata should be carefully selected and constrained to the minimal information necessary to interpret what the wrapper is encapsulating.
- **Distribution** is an everlasting challenge for researchers, who are constantly dealing with data that are located in distinct geographic locations and most of the times they are accessing these data from different computers. It is very important to prepare systems for future interoperability, enabling transparent access to distributed resources and easing remote access.

3.2.2 Service-Oriented Architectures

Web services are, nowadays, the most widely used technology for the development of distributed applications [122]. The World Wide Web Consortium (W3C) defines web services as "software system designed to support interoperable machine-to-machine interaction over a network" [123]. This wide definition allows us to consider a web service as any kind of Internet available service as long as it enables machine-to-machine interoperability.

Service-Oriented Architectures (SOA) is a modern application deployment architectural style. The rationale behind these architectures is that what the applications are connecting are services and not other applications. Considering that every component that applications require can be seen as an independent service, we create an implementation and deployment strategy based in this paradigm. In traditional web application architectures, the deployment can be decomposed in three generic layers: the presentation layer, the business logic layer and the data access layer. In SOA architectures, the idea of focused architecture layers does not exist. That is, each architecture component is independent from the remaining and by combining miscellaneous components we can compose multiple applications. This empowers two main concepts: reusable software, which are applications wrapped as services that can be used in a multitude of applications, and service composition, where applications can be built by combining sets of services.

Various patterns have been exploited over the years to promote the creation of service-oriented architectures. Natural selection left two key standards that act as the foundation for new applications with advanced service composition methods. SOAP is the strictest standard, involving a more complex set of technologies to control all steps of the composition. Unlike SOAP, REST is a more open alternative, leaving the standardization and data exchange validation to the superior application layer.

SOAP

Standardized web services have the main purpose of providing a unified data access interface and a constant data model of the data sources. Simple Object Access Protocol (SOAP) [124], Universal Description, Discovery and Integration (UDDI) [125] and Web Services Description Language (WSDL) [126] are the currently used standards and they define machine-to-machine interoperability at all levels, ranging from the data transport protocol to the query languages used. Web service interoperation occurs among three different entities: the service requester, the service broker and the service provider.

SOAP standard defines a comprehensive architecture based on several layers where all the components required for a basic message exchange framework are defined. These components include message format, message exchange patterns, and message processing models, HTTP transport protocol bindings and protocol extensibility.

WSDL standardizes the description of web service endpoints. This description enables automation of communication processes by documenting every element involved in the interaction (from the entities to the exchanged messages). The definition encompasses several components that are structured in order to facilitate communication with other machines and ease the readability of the web service by humans. Obviously, thinking of a complex web service, we realize that there are numerous data types that need to be described. Whether we are dealing with person’s names or protein interactions, each scenario has its respective internal model, which must be exposed to make the system interoperable. WSDL recognizes this need and can use XML Schema Definition (XSD) as its canonical type system. Despite this, we cannot expect that this grammar will cover all possible data types and message formats in the future. To overcome this issue, WSDL is extensible, allowing the addition of novel protocols, data formats or structures to existing messages, operations or endpoints [127].

UDDI allows describing, discovering and managing in the web services environment. UDDI usually offer a central registry with "publish and subscribe" features that allows the storage of service descriptions and detailed technical specifications about the web services. The storage mechanism relies once again on XML to define a metadata schema that can be easily searched by any discovery application

REST

REST service architecture styles are nowadays the most widely used solution for interoperable software. REST-based solutions are used in every modern application form, enabling streamlined data exchanges for all kinds of fields. REST stands for representational state transfer and this service architecture uses all valuable features inherent to the HTTP protocol [128]. Starting with the representation of resources with unique URLs, developers can employ the GET, POST, PUT and DELETE operations available in the HTTP protocol to enhance the access to remote data. Whereas normal web pages are designed for human consumption, meaning that browsers only use the GET command to load data in a closed window, the REST style uses all available operations to enable machine-to-machine communication.

Developers can configure this page to respond with HTML, XML, JSON, CSV or, most simply, free text. Using REST web services it does not matter what is the response structure and its inner format, the essential requirement is that the exchanged messages are understood between both the intervenient in the exchange. This feature makes REST web services a lightweight and highly customizable approach for exchanges between machines [129]. In April 2012, the Programmable Web library[11] contains 5551 web APIs, 3740 of which are REST-based, a staggering 78%.

This steady use growth and its availability in all kinds of services, from Twitter to UniProt, is making REST the de facto solution for modern service oriented architectures, empowering a true web-as-a-platform environment.

[11] http://www.programmableweb.com

The Promise and Limits of SOA

Service composition strategies are essential for achieving integration and interoperability. We need an architecture that delivers organic interactions. This means that the composition of services should be provided naturally, responding to project demands swiftly and enabling the continuous improvement of available software without disrupting existing systems. The resulting ecosystem, with greater flexibility, improved scalability and better responsiveness, can be tailored to any modern development scenario.

However, these benefits came with a cost. In complex service composition scenarios, involving multiple interacting workflows or mashups, many distinct problems may arise. Distributed architectures may suffer from connectivity issues. With the Internet as the main communication line, it is vital that a web connection is present 24/7 in the client applications. For instance, developing mobile applications that rely on a remote service for their execution limits the application usability to realities where the Internet is available.

Performance and robustness may also suffer. On the one hand, data exchanges can be lighter and therefore faster, due to the use of smaller objects. On the other hand, the parallelization of service execution is not straightforward, meaning that workflows' services need to wait for the previous service completion before moving on. The lack of robustness is exemplified by taking down a single service in a service-oriented system. This can cause the entire system to fall apart, even if 99% of the involved components are operational.

Managing a collection of distributed services is also challenging for SOA. These require additional metadata to maintain adequate governance and application ownership. SOA blurs the boundaries of where a self-contained application starts and ends. With the emergent domination of the mobile application market, it is fairly trivial to write applications that simply provide another view over a pre-existent service, without further added value.

The limit of SOA tends to the concept of "everything-as-a-service". This new paradigm requires efforts for both service consumers and producers. Whether we are simply requesting a listing from movies database, or delivering access to our oral cavity knowledge base, we need to reengineer existing systems and change the way we think about software systems' architectures.

3.2.3 Heterogeneity

Online resource heterogeneity issues are of growing relevance, triggered by the constant evolution of the Internet and the increased facility in publishing content online. We can classify online resource heterogeneity in five distinct groups, varying according to their complexity and their involvement at hardware/software levels (Table 3-1).

Table 3-1. Content heterogeneity organization according to hardware/software dependencies and complexity, from physical data heterogeneity to the variety of access methods.

PHYSICAL DATA	LOGICAL STORING	DATA FORMATS	DATA MODELS	ACCESS METHODS
→ Web Server	→ Relational Database	→ HTML	→ Structure	→ Local Access
→ FTP Server	→ OO Database	→ CSV	→ Ontology	→ Remote APIs
→ File Server	→ Text File	→ XML	→ Semantics	→ Web Services
→ Backup Tape	→ Binary File	→ TXT		
		→ Excel		

Hardware related issues arise when dealing with physical data storage. For instance, a medical imaging information system may require that image backups, stored in tapes, be integrated in the system as well as images in the main facility web server. In this scenario, the integration setup is considerably complex.

When dealing with file access in any storage, we may have logical storing heterogeneity. Content can be stored in a relational database, a simple text file or a binary file, among others. Hence, these several formats are accessed with entirely different interfaces. For instance, integrating data from a Microsoft SQL Server 2008 database is a completely different process from reading a binary file from a FTP server.

The next level where heterogeneity can be a problem is at data format levels. Data stored in the same physical format can be stored in a distinct syntax. Although programming languages' evolution has improved access to distinct file formats, reading a simple text file or a HTML file are operations that require different strategies and methods. A simple scenario could be the integration of several accounting results, which are offered in CSV formats, Excel files and tabular text files. To successfully integrate these files, developers must implement distinct access methods to the three logical formats.

Moving deeper in the software layer, we reach the data models level where heterogeneity issues arise when files are distinctly structured or do not obey to the same schema/ontology. Difficulties in solving this issue were greatly reduced with the appearance of the XML standard, and most of the modern applications rely on this standard. Despite having normalized the process of reading and storing information, XML allows an infinite number of valid distinct structures, which are different from application to application. Once again, heterogeneity has to be solved with information and relation mappings that can correctly transpose information structured following ontology A to ontology B. These mappings are quite complex and traditionally require some kind of human effort for success.

Finally there is access methods heterogeneity. Web services have evolved and are the primary method for remote data access with standard protocols and data exchange formats. Nevertheless, web services may be divided in HTTP-based (REST or SOAP) and XMPP web services, further increasing the complexity of managing distinct interoperability requirements within a single platform.

Summarily, resource heterogeneity is the main challenge for the development of novel information integrative platforms. For the creation of novel knowledge and information management systems, the integration of distributed and heterogeneous data are a major drawback, reducing future interoperability features.

3.3 Modern Data Integration

As mentioned in Chapter 2, one of the great caveats for the successful integration of data in the life sciences domain is the constantly growing amount of data and, consequently, of ways to access those data. This is not true solely for bioinformatics. Whereas in the early 21st century it was very difficult to find ways to publish our personal content online, the evolution and widespread availability of Internet access technologies has leveraged an explosive growth of data in all fields.

3.3.1 Resource Integration Strategies

To deal with resource heterogeneity issues or to simply centralize large amounts of distributed data in a single system, researchers have to develop state of the art resource integration architectures. The main goal of any integration architecture is to support a unified set of features working over disparate and heterogeneous applications. The heterogeneity may be located at the previously presented levels, which include software and hardware platforms, diversity of architectural styles and paradigms, content security issues or geographic location. In addition to these technical restrictions to integration, there are also other hindrances such as enterprise/academic boundaries or political/ethical issues. Whether we are simply dealing with the integration of a set of XML files or with distributed instances of similar databases, the concept of resource integration will generically rely on hard-coded coordination methods to centralize the distributed information or to give the idea that the data are centralized.

Several strategies for data integration can be used - Figure 3-2. These approaches differ mostly on the amount and kind of data that is merged in the central database. Different architectures will also generate a different impact on the application performance and efficiency.

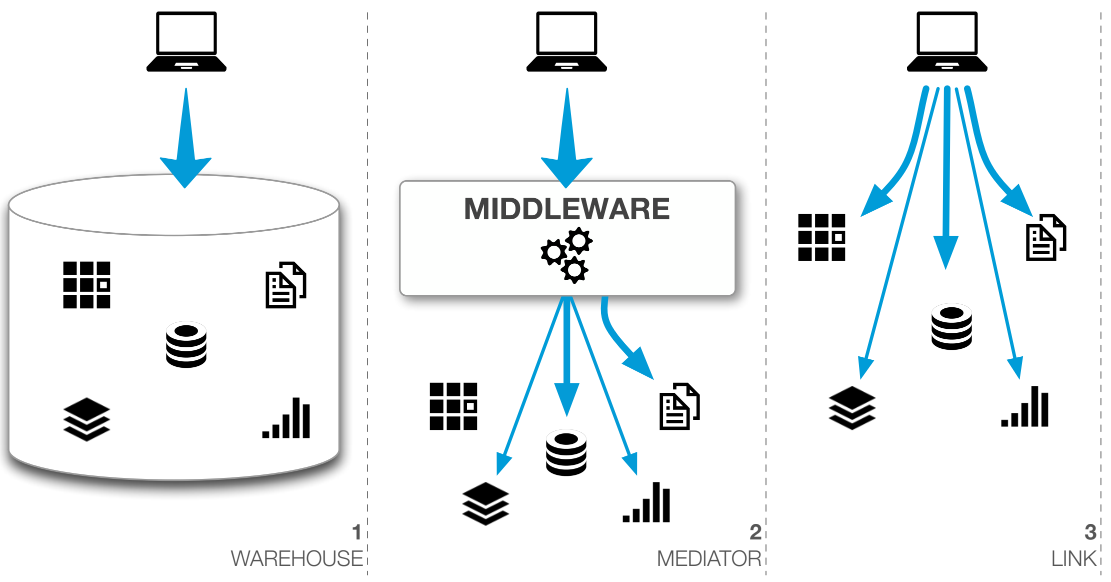


Figure 3-2. Data integration models categorized according to their relation with the integration application and the integrated online resources. 1) Warehouse integration replicates entire datasets from external resources in a new knowledge base. 2) Mediator-based solutions rely on a middleware layer to serve as a proxy for connections to a set of virtually integrated external resources. 3) Lightweight link-based integration is based on direct connections to the external distributed resources.

Warehouse solutions (Figure 3-2.1) consist on the creation of a large database that contains data gathered from several resources. The central database – the warehouse – may consist of a mesh of connected repositories that the data access layer sees as a single database. In terms of implementation, this model requires that mappings are made from each data source to the central warehouse data model. Next, the content is moved entirely from its source to the new location. The final result is a new data warehouse where the content from the integrated data sources is completely replicated.

This model raises several problems in terms of scalability and flexibility: warehouses' size can grow exponentially and each database requires its own integration schema. This means that for each distinct database, developers have to create a new set of integration methods, resulting in a very rigid platform. Despite these issues, this technique is very mature and a considerable amount of work has already been done to improve warehouse architectures. Nowadays, the debate is focused on enhancing warehouse integration techniques [130] and solving old problems with state-of-the-art technologies [131, 132].

Another widespread strategy involves the development of mediators – a middleware layer – connecting the application to the data sources - Figure 3-2.2. This middleware layer enables a dynamic customization of user queries performed in the centralized entry point, extending their scope to several databases previously modelled in a new virtually larger database. Kiani and Shiri describe these solutions [133] and a good example can be DiscoveryLink [134]. Mediator-based solutions are usually constrained by data processing delays: they require real-time data gathering, which can be bottlenecked by the original data source. Additionally, the gathered content also has to be processed to fit in the presentation model, hence, compromising even more the overall efficiency of the system.

Finally, link-based resource integration consists of aggregating in a single platform links to several relevant resources throughout the Web (Figure 3-2.3). This is the most widely used integration model due to the simplicity of collecting and showing links related to a certain subject. However, inherent in this simplicity are several drawbacks, especially regarding the limitations imposed by the fact that there is no real access to data, only to their public URLs. Most of the modern resources are dynamic which means that access to content may be generated in real-time. Also, in the scientific research area, there are new data emerging daily. Therefore, the system requires constant maintenance in order to keep the system updated with the area novelties.

Despite the fact that these approaches cover almost all possible solutions for data integration, there are many problems that have not yet been solved. After a careful analysis of these models we can conclude that, for the majority of life sciences resource integration scenarios, the best option is to create a hybrid solution that is capable of coping with the main disadvantages of the three strategies and take advantage of their main benefits as well. The development of hybrid approaches has gained momentum in the recent years especially with the introduction of novel data access techniques like remote and web services.

In spite of the strategy chosen to integrate a collection of heterogeneous resources, there are several concerns that should be taken in account: application coupling, intrusiveness, technology selection, data format and remote communication [135]:

- **Application coupling** enforces the good software development practice of "low coupling, high cohesion", an ideal that is also applicable to integration strategies [136]. High coupling results in high application dependencies on each other, reducing the possibilities of the applications evolving individually without affecting other applications. The optimal results would be resource integration interfaces that are specific enough to implement the desired features and generic enough to allow the implementation of changes as needed.
- **Intrusiveness** should be one of the main concerns when developing integration applications. The integration process should not impose any modifications in the constituent applications. That is, the integration strategy should operate without any interference in the existing applications and both the integrator and the integrated application should be completely independent.
- **Technology selection** is cumbersome when combining environments where multiple distinct applications interact with each other. Despite being dissimulated as local interactions by the integration engine, these remote interactions, available in the majority of programming languages, are very different due to the resort to network capabilities. Remote communication concerns are reduced with the adoption of asynchronous communication techniques and the support for communication error solving, thus reducing network error susceptibility.
- **Data formats** must be unified in integrative applications. Traditionally, this requirement is impossible to fulfil due to the fact that some of the integrated data sources are closed or considered legacy. In these scenarios, the solution consists in creating a translator that maps the distinct data formats to a single model. In this case, issues may arise when data formats evolve or are extended.

The implicit complexities that arise when dealing with online resource integration require large efforts and expertise to be overcome. Like any other issue, integration is firmly related with the scientific area in question and, with this in mind, the adopted strategy or model must take in account several variables present in the environment where it will be implemented.

3.4 Towards Software Interoperability

Interoperable software is the main enabler of modern informatics systems. Whereas in integration we deal with the development of a unified system that includes the features of its constituent parts, interoperability deals with single software entities that can be easily deployed in future environments. This means that interoperability is a software feature that facilitates integration and collaboration with other applications. The ISO/IEC 2382-01 defines interoperability as follows: "The capability to communicate, execute programs, or transfer data among various functional units in a manner that requires the user to have little or no knowledge of the unique characteristics of those units"[12].

In a sense, developing interoperable software means developing open and future-proof applications. Nowadays, stakeholders are no longer interested in creating closed tools. They are geared towards the creation of an open application ecosystem where any external developer is able to connect to and exchange data with the central system.

3.4.1 The axioms of Interoperability in Informatics

Interoperable systems can access and use parts of other systems, exchange content with other systems and communicate using predefined protocols that are common to both systems. This interoperability can be achieved at several distinct levels as pointed by Tolk's work (Figure 3-3). For our research work, the essential levels are the ones that encompass syntactic and semantic interoperability.

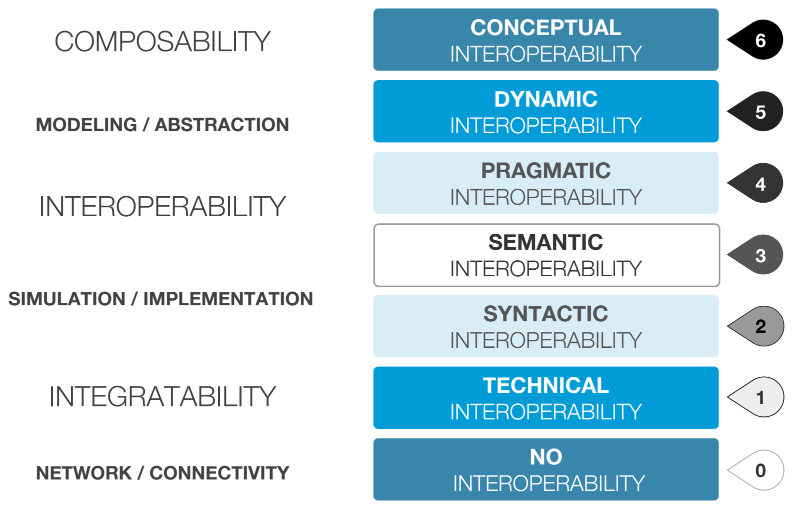

COMPOSABILITY
MODELING / ABSTRACTION
INTEROPERABILITY
SIMULATION / IMPLEMENTATION
INTEGRATABILITY
NETWORK / CONNECTIVITY
CONCEPTUAL INTEROPERABILITY
6
DYNAMIC INTEROPERABILITY
5
PRAGMATIC INTEROPERABILITY
4
SEMANTIC INTEROPERABILITY
3
SYNTACTIC INTEROPERABILITY
2
TECHNICAL INTEROPERABILITY
1
NO INTEROPERABILITY
0

Figure 3-3. Levels of conceptual interoperability model defined by Tolk [137]. Each level (from 0 to 6) increases the available interoperability features, from no interoperability to full software composability.

[12] http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=7229

Software syntactic interoperability can be defined as the characteristic that defines where multiple software components can interact regardless of their implementation language or software/hardware platform. Syntactic software interoperability may be achieved with data type and specification level interoperability. Data type interoperability consists in distributed and distinct programs supporting structured content exchanges whether through indirect methods – writing in the same file – or direct methods – API invoked inside a computer or through a network. Specification level interoperability encapsulates knowledge representation differences when dealing with abstract data types, thus, enabling programs to communicate at higher levels of abstraction – web service level for instance.

Semantics is a term that usually refers to the meaning of things. In practice, semantic metadata are used to specify the concrete description of entities. These descriptions and their relevance are detailed further in this document. Summarily, they intend to provide contextual details about entities: their nature, their purpose or their behaviour among others. Hence, semantic software interoperability represents the ability for two or more distinct software applications to exchange information and understand the meaning of that information accurately, automatically and dynamically. Semantic interoperability must be prepared in advance, in design time and with the purpose of predicting behaviour and structure of the interoperable entities.

3.4.2 Foundations for Software Interoperability

When working with service composition we are required to enforce interoperability. Composing services that already support some high degree of interoperability eases developments but it is not a mandatory requirement. As long as there is an open door to share data or features, it is up to the interested party to implement some kind of wrapping middleware layer.

Integration and interoperability are directly tied together. There is no integration without interoperability and interoperability is driven by the need to integrate. From a centralized perspective, we can define integration as inward interoperability, to get resources from an external physical or logical location into our own unifying resource. Nevertheless, true software interoperability is only present when there is also outward interoperability. This means that our system should not only integrate data, but should also expose it for external usage. Assessing service composition as a whole, we want our resources to be interoperable so that others can integrate them, enabling a multidimensional data flow.

This focus on the interoperability of data and software has leveraged the birth of novel development paradigms. Semantic Web technologies emerge as a modern solution tailored to solve interoperability issues in any research field. With new standards for describing, publishing and accessing data, the semantic web paradigm appears as the most viable alternative for implementing best-of-breed service composition strategies towards a better integration of resources and the improved interoperability amongst distributed autonomous knowledge bases.

3.5 The Semantic Web

The dramatic growth in content promoted by recent web developments like Web2.0 and social tools, combined with the ease in the publication of online content, have the major drawback of increasing the complexity of resource description tasks. Standard web technologies cannot support this exponential increase.

Tim Berners-Lee, the self-proclaimed inventor of the modern Internet and director of W3C, promoted semantic Web developments in 2001 [138]. His futuristic initiative envisaged to smoothly link personal information management, enterprise application integration and worldwide sharing of knowledge. Therefore, tools and protocols were developed to facilitate the creation of machine-understandable resources and to publish this new semantically described resources online. The long-term purpose was to make the Web a place where resources are shared and processed by both humans and machines, enabling computers to comprehend, navigate, manipulate and infer reasoning from the Web of data. The W3C Semantic Web Activity group has already launched a series of protocols to promote the developments in this area – Figure 3-4.

Adding semantic features to existing content involves the creation of a new level of metadata about the resource [139, 140]. This new layer will allow an effective use of described data by machines based on the semantic information that describes it. These metadata must identify, describe and represent the original data in a universal and machine understandable way.

In order to make the semantic web possible, developers and researchers need to cooperate. It is important to create and broadcast centralized ontologies for several public interest areas and to empower the adoption of these ontologies by research groups and private companies. Nevertheless, this crucial step can only be given if the cooperation efforts originate enhanced semantic technologies that ease the complex task of describing content. Research groups working with state-of-the-art technologies must promote this difficult step that will require deep changes in the developed applications. Only promoting this use we can foster the development of a new, cleverer, Internet [141, 142].

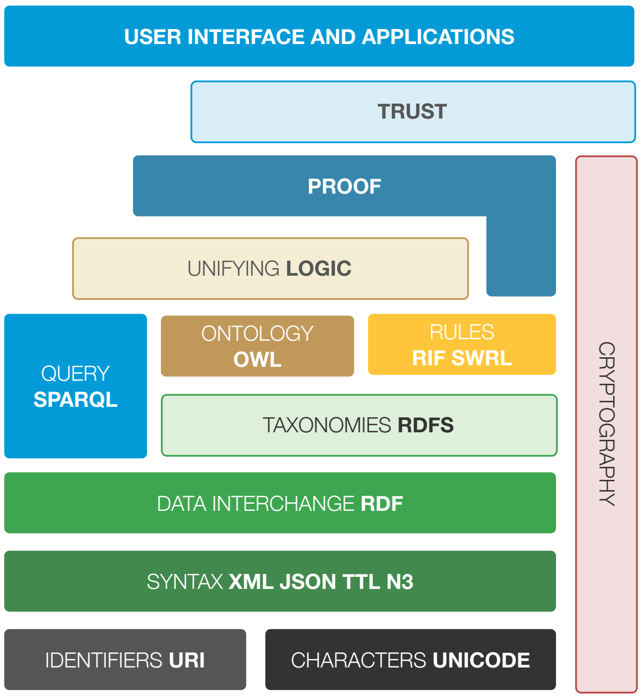


Figure 3-4. Semantic Web Stack, from URI resource identification to the top-level applications.

3.5.1 Expressing Knowledge

The fundamental building block of semantic web knowledge is a statement. Whilst this may be an oversimplification, it opens many possibilities for setting the semantic web as a standard for integration and interoperability. Whereas traditional WWW content is built for human consumption, semantic web content consists primarily of statements for application consumption. Whereas in traditional WWW it was up to users to decide and acknowledge the connections amongst data bits, semantic web statements are linked together through constructs that form the meaning of a link.

It is in these relationships that resides the semantic web's added value. By connecting millions of statements together we can form a rich knowledge base, even if the information remains distributed. Semantic web relationships may be definitions, associations, aggregations, and restrictions, amongst other hybrid custom connections. These relationships are better understood as a graph, as shown in Figure 3-5, listing a small set of statements.

Statements and established relationships define concepts, for example a Person has a name, and instances, such as “Alice is a friend of Bob” (p:hasFriend). The set of statements defining concept relationship form the ontology. Likewise, the ones referring to unique individuals form the data. Any statement can be asserted, created by a direct connection, or inferred, discovered by using additional logic.

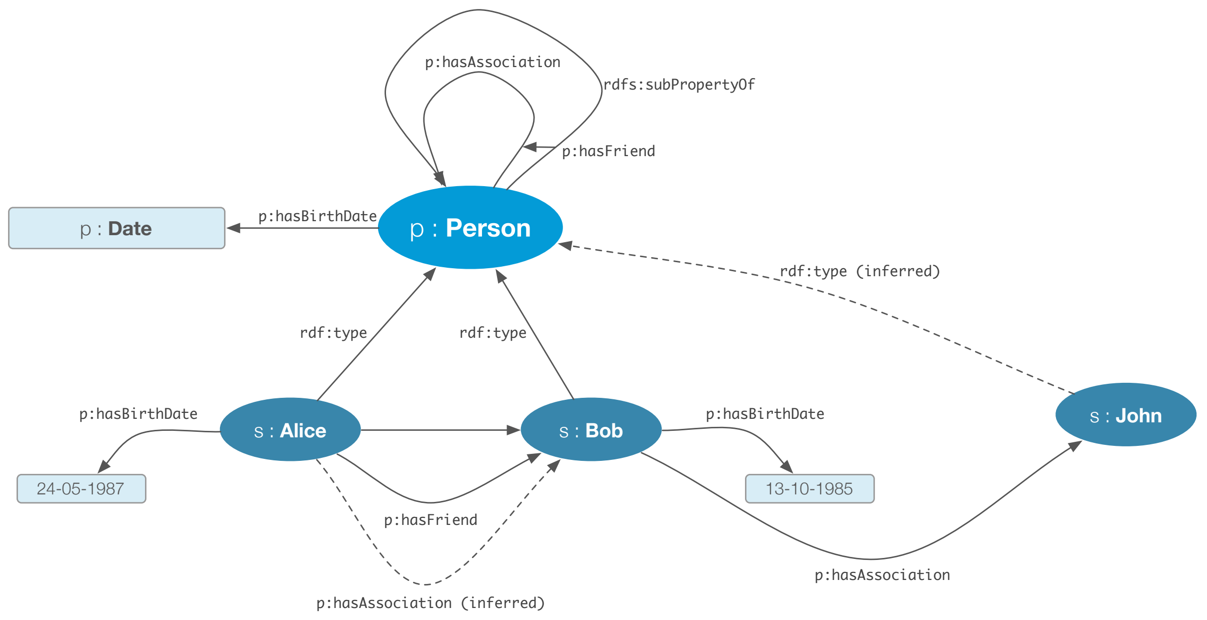


Figure 3-5. Example RDF graph, highlight direct and inferred relationships within a common object-oriented Person ontology.

From Figure 3-5 graph we can extract a set of statements formed by three elementary components - **subject**, **predicate**, and **object**. These individual triples are what forms knowledge in the semantic web. The subject is the element of what we are saying something new, the predicate is the meaning of the relationship we are establishing, and the object is what we are explicitly stating about the subject. To express these statement sets, we need to identify resources uniquely and rely on a group of technologies to integrate information and explore acquired knowledge. This is achieved through the combination of four web protocols: URI [143], RDF [144], OWL [145] and SPARQL [146, 147].

Identifying Resources

The Uniform Resource Identifier (URI) is a simple and generic standard that proposes a sequence of characters to enable the uniform identification of any resource across the entire Internet. The “Resource” term is used in a general sense as it can identify any kind of component. URIs can identify electronic documents, services, data sources and other resources that cannot be access via Internet like humans or corporations. Identifier refers to the operation of unequivocally distinguish what is being identified from any other element in the scope of identification. This means that we are able to distinguish one resource from all other resources regardless the working area or resource purpose.

Using URIs allows the definition of namespaces that can be expanded to accommodate new data or features, enabling a scalable access to objects. In a trivial example, **http://www.facebook.com/** can be reserved as a generic namespace for Facebook data and, in this case, **http://www.facebook.com/<username>** uniquely identifies a user within Facebook's context. Similarly, we can identify a protein in UniProt's knowledge base through the combination of UniProt's namespace with the protein accession number, resulting in **http://www.uniprot.org/uniprot/P51587**.

Describing Resources

The description of resources should provide details about its nature, intent or behaviour as well as being, generically, "data about data". The Resource Description Format (RDF) is designed as a protocol to enable the description of web resources in a simple fashion [148]. The syntax neutral data model is based on the representation of predicates and their values. A resource can be anything that is correctly referenced by an URI and is currently, like the latter, not limited to describing web resources.

A major advantage for using RDF as a knowledge storage facility is its proneness to sharing. The simple triple/graph structure is optimal for data exchanges. RDF has no default data format and is not restricted to constrained data models. When all standards are respected, combining two RDF graphs is easier than integrating two databases or merging a couple XML files.

The basic idea of storing data as triples is by itself a powerful tool for the integration of data. RDF data does not require any translation, mapping or contextual information to be used. That is, by storing data in our knowledge base as triples, we are limitless to explore that data and connect it with external resources without the traditional integration and interoperability problems we face while integrating information from relational databases, CSV or XML files.

Ontologies

Any scientific research field deals with specific terminology that is associated with a particular area. Ontology defines the collection of terms and relations between terms that are more adequate for a given topic [149]. These relationships, often designated axioms, establish connections between terms in the thesaurus that mimic the real world.

Ontology is the collection of consensual and shared models in an executable form of concepts, relations and their constraints tied to a scaffold of taxonomies. In practical terms, we use ontologies to assert facts about resources described in RDF and referenced by an URI. The Web Ontology Language (OWL) is the *de facto* ontology standard, extending the RDF schema.

Querying Resources

To query RDF files and, in a larger scale, the Semantic Web, W3C developed the SPARQL Protocol and RDF Query Language syntax. SPARQL is an SQL-like query language that acts as a friendly interface to RDF information. SPARQL queries are directed to an endpoint, a service that accepts queries and outputs the results in different formats. Besides being the semantic web query language, SPARQL is also the protocol for setting up HTTP connections from SPARQL clients to SPARQL endpoints.

SPARQL syntax is very similar to SQL and enables four distinct five types: **SELECT**, **CONSTRUCT**, **ASK**, **UPDATE** and **DESCRIBE**. **SELECT** statements are similar to SQL selections where we bind variables in our query to the results we expect to obtain from the knowledge base. **CONSTRUCT** queries enable the addition of new graphs or the transformation of existing triple graphs into new datasets. **ASK** queries are used to evaluate the existence of a particular resource or relationship. The result for **ASK** queries is always a boolean value, true or false. **UPDATE** queries are used to issue updates to data already existing in a knowledge base graph. At last, the **DESCRIBE** query, limited to a single resource, returns all known relationships for the given resource. This is an essential feature for the automated discovery of new data without any awareness of existing structures. **DESCRIBE** queries power up the LinkedData guidelines.

The SPARQL language is complete with advanced data access features just like SQL. Filters, sorting, limits or modifiers are all present and are complemented with a clean variable binding schema, making SPARQL an advanced query language.

3.5.2 LinkedData

Despite the increased awareness regarding semantic web potential for integration and interoperability, its adoption has been painfully slow. “Semantic creep” is the new term for this late adoption, sustained mostly by social obstacles and the proverbial chicken and egg problem: the lack of immediate advantages after adopting the semantic web paradigm is only resolved when a critical mass of cross-linked knowledge bases also does so [53].

The overarching goal of the semantic web is to enable a truly distributed knowledge network. To connect the wealth of ontologies and data widespread in this web of structured data, a set of best practices - LinkedData - were proposed by Tim Berners-Lee more recently [58]. By adopting these guidelines, knowledge bases are exposed in a standard, well-defined fashion, enabling a richer semantic web-based navigation [60]. LinkedData imposes four key rules:

- Use URIs as names for things
- Use HTTP URIs so that people can look up those names.
- When someone looks up a URI, provide useful information, using the standards (RDF*, SPARQL)
- Include links to other URIs so that they can discover more things.

Despite being somewhat vague, obeying these rules empowers an easily navigable distributed data graph. With the increased awareness surrounding semantic web technologies, the LinkedData space already accounts for billions of assertions, all in a single open data space (Figure 3-6) [59]. This openness is a defining feature for LinkedData. With an adequate exploitation of semantic web's strategies, we can correlate and connect data from entirely distinct fields, from media, publications, geography, life sciences or the corporate domain.

LinkedData obeys the semantic web principles and whilst most of the data navigation should be made autonomously by intelligent software systems, we can also query LinkedData using SPARQL [150]. This is federated query processing at its best, where data to complete each query is dynamically discovered and virtually integrated in real-time.

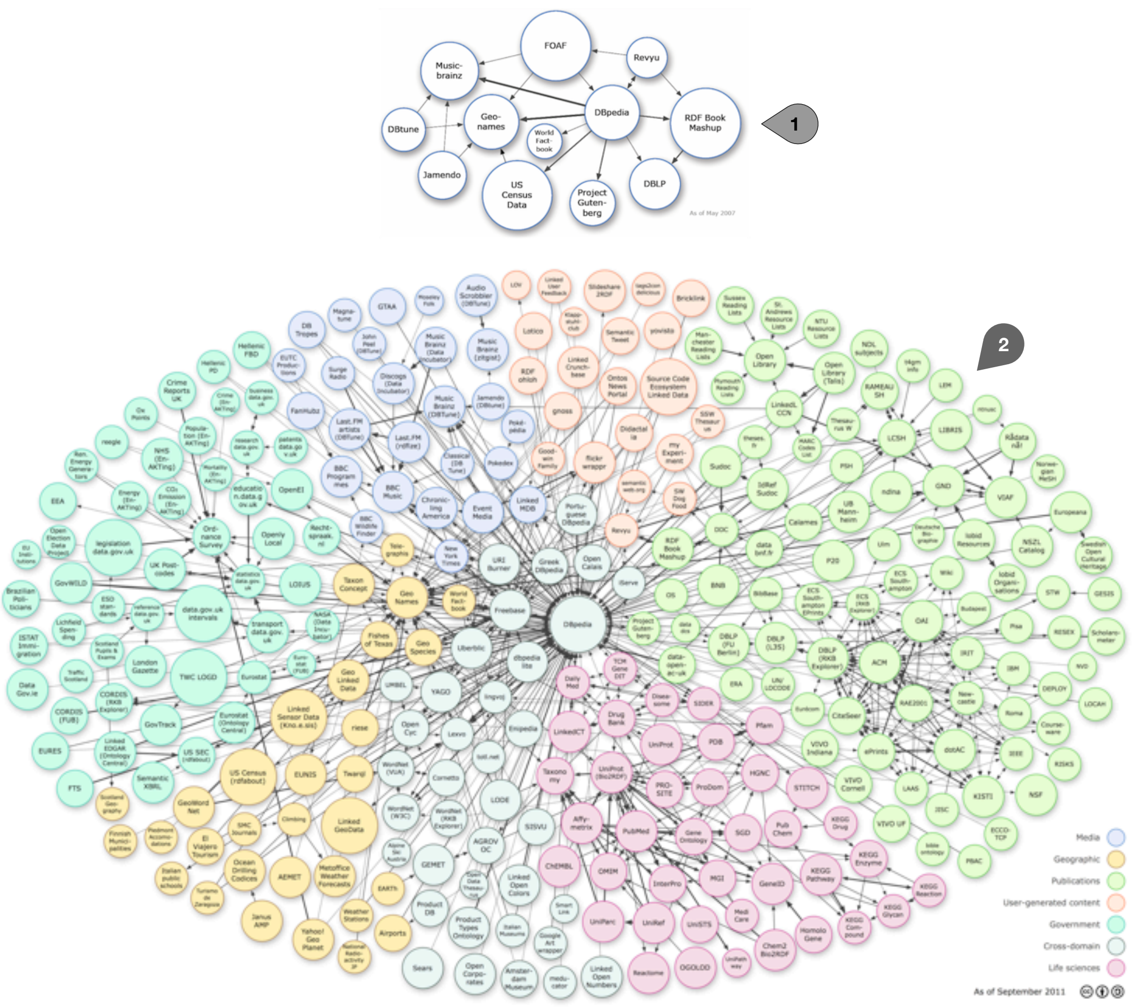


Figure 3-6. LinkedData cloud evolution, from 2007 (1) to 2011 (2)[13]**.**

3.6 Discussion

3.6.1 Shortcomings and Challenges

Researchers' daily work is getting more complex as traditional simple tasks like locating necessary information, gathering it and working with tools to process it get more difficult. The growing number of software tools is not helpful as well. Despite their quantity, their quality is questionable; each tool works differently and requires distinct end-user skills. Along with application complexity, there is also the immense number of data formats. In a single scientific area there are numerous resources, applications, services, data formats, data models and data types to consider. This imposes time consuming tasks, like manual data transformations or development of custom wrappers and converters, which are far beyond scientific researchers' scope.

[13] **Linking Open Data cloud diagram**, by Richard Cyganiak and Anja Jentzsch. http://lod-cloud.net/

Whilst there are already various notable efforts to overcome these issues, there is still a clear lack of service depth. New tools and knowledge bases are starting to include web services by default. However, this is causing further problems as these are shallow APIs, they are not normalized and do not adhere to any previous guideline. The problem itself is not the absence of standardization formats for data exchanges and services, but their lack of use.

3.6.2 An Emerging Architecture Trend

As trivial as it may seem, there is no denying that the complex service architectures at play in existing software systems occupy a determinant role. The mobile web itself, especially with the omnipresent "apps", relies on service-based technologies to operate. Whether we are checking our emails, sharing a photo with our friends or reading the latest news, services empower our connections to the online world.

With these strategies, modern software engineering trends emerged. Nowadays, developers do not think about building applications for single use or targeting a unique scenario, developers must build entire platforms. In fact, more important than being a unique popular application, the success of novel systems resides in the ability to provide platforms that others can explore, enabling the appearance of a new controlled ecosystem. The most widely used modern trend is to deploy a central logical ICT infrastructure to support a multitude of applications, targeting distinct operating systems, hardware platforms and users (Figure 3-7).

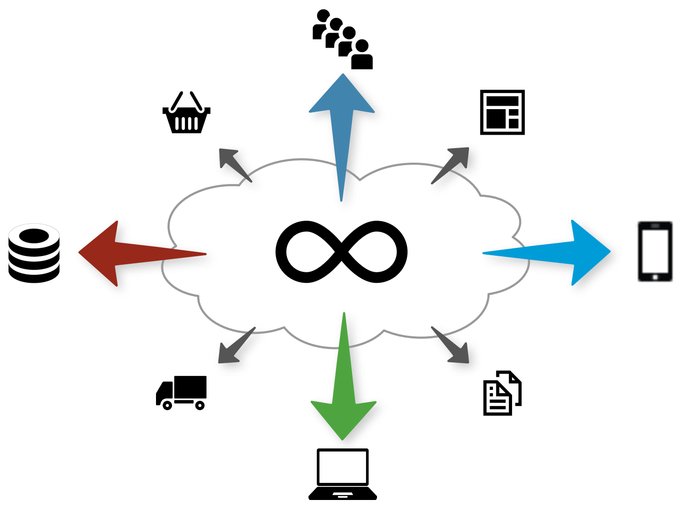


Figure 3-7. Modern software platform architecture trend with a centralized (cloud-based) source providing data and features to miscellaneous client applications, each targeting a distinct market segment, hardware device or operating system.

A quick overview on the approaches adopted by Apple, Amazon, Evernote, Facebook, Google or Twitter, highlights this strategy. The idea of creating a platform that any developer can rely on to build new applications is as enticing as writing HTML was in the early WWW days. These companies adopted a cross-product API strategy that changes the way new applications are built. Not only do they provide a collection of APIs to external developers, but they also promote the use of these services internally in their products. Completed with a rich documentation set, these companies build their brands capitalizing on the community interest in their products and the community skills to improve the existing ecosystem.

For software engineers, these new architectures push forward new demands for the integration of existing resources and the interoperability with new external systems.

3.6.3 The Next Step for Service Composition

This chapter provides an overview over the various technologies and paradigms used to empower a new software engineering age supported by service composition strategies. Along with this description, there is also the highlight of where we can go in the future using integration and interoperability demands as a driving guideline for software development.

Software engineering dynamics and adaptability are remarkable. Using similar architectures we can devise new platforms for controlling car manufacturing pipelines or next-generation sequencing hardware, for entertainment or bioinformatics. This latter field is definitely an innovation driver, demanding the employment of best-of-breed techniques to solve computational biology problems. Whether re-engineering existing technologies or using modern semantic web developments, state-of-the-art solutions must be transposed from the general computer science field to bioinformatics.

For this matter, we need to go beyond service composition in bioinformatics, taking it one step further. Integration and interoperability requirements define architectures for novel biomedical applications and all stakeholders should promote the adoption of these ideals. As active players in the biomedical software community, we introduce a set of contributions aiming to enhance the wide field of service composition for biomedical applications, and to thrive under the immense bioinformatics opportunities.

4. CONTRIBUTIONS TO WORKFLOW-BASED SERVICE COMPOSITION

"I do not fear computers. I fear the lack of them."
- Isaac Asimov

With the latest advances in software architectures, modern service composition strategies empower new tactics for interoperability. Emerging service-oriented architectures and software-as-a-service trends bring new approaches that are more scalable, flexible and efficient. Therefore, new applications and services ease researchers' investigation tasks, enabling advanced interactions amongst remote applications through distributed workflows.

This chapter introduces our contributions to the research on workflow-based service composition within the context of the European EU-ADR Project[14]. The goals behind this project concern the improvement of post-marketing pharmacovigilance strategies and technologies. Contemporary disease treatment and prevention revolves around a dynamic medication market, where a myriad of pharmaceutical companies research, develop and introduce numerous drugs in the international health marketplace. Hence, drug safety continues to be a major concern for worldwide policy stakeholders as it continues to injure patient's health and, in many cases, lead to increased mortality risk.

Within the EU-ADR scenario, several features and research results are provided through web services. This propelled the creation of a new platform, the EU-ADR Web Platform, where services play together to form a set of innovative pharmacovigilance workflows. The development of this platform leveraged miscellaneous service composition challenges, namely on defining interoperability standards and wrapping Taverna workflows. From the creation of custom drug studies to the remote execution of signal analysis workflows up to

[14] http://euadr-project.org/

cross-analysis against millions of anonymous electronic health records [151, 152], the EU-ADR Web Platform enables an insightful exploration of pharmacovigilance signals' evolution resulting in a superior risk evaluation. The EU-ADR Web Platform is available online at http://bioinformatics.ua.pt/euadr/.

4.1 Delivering Advanced Pharmacovigilance Workflows

The traditional pharmacovigilance approach tackles the problem from a pre-market perspective, conditioning drug approval. Both the European Medicines Agency (EMA) and the US Food and Drug Association (FDA) established rigorous guidelines for new medicine approval, requiring intense testing, which results in a long and complex lab-to-market development cycle. Along with these guidelines, pharmaceutical companies must also define thorough risk management plans for post-market drug stages.

Consequently, the relevance of post-market pharmacovigilance in the health domain has been growing steadily over the last four decades. Research in this area involves the exploration and assessment of signals, defined by the World Health Organization as undisclosed assertions on direct relationships between adverse events, such as gastro-intestinal bleeding, and a drug, like rofecoxib [153]. Clinicians use spontaneous reporting systems to identify adverse drug reactions [154]. Despite this, there is high-demand for novel software tools capable of improving the post-marketing drug monitoring workflow [155]. By taking advantage of modern knowledge engineering technologies, developers are able to overcome the limitations associated with insufficient clinical trial data, complex monitoring statistics and closed general practice data silos. Text- and data-mining tools, combined with service composition strategies, pave the way for enhanced *in silico* signal identification and adverse drug reaction assessment [156].

Whilst these software products are already available, their use is limited to a small group of technologically skilled research experts. Hence, the creation of the EU-ADR Web Platform to tackle these challenges, extending existing tools availability to every researcher, clinician or stakeholder, through a web-based pharmacovigilance suite.

4.1.1 21st Century Pharmacovigilance

Hårmark and Grootheest research explains pharmacovigilance underlying concerns with current drug evaluation approaches [157]. Whilst drug safety concerns are becoming more prominent, the lack of adequate software to correctly understand drug adverse reactions continues to challenge the pharmaceutical industry and research community.

The risk associated with any marketed drug triggers critical safety concerns, which, in their turn, leverage a constant revision and update of medical products' information. For these tasks, modern adverse drug reaction (ADR) monitoring becomes essential. Despite the complex set of drug trials, including the famous final randomized double blind evaluation, clinical trials data are in most scenarios insufficient to assess drug risk. Rare ADRs, ADRs identified in particular population cohorts or ADRs with long latency, require intensive post-marketing drug analysis.

At this stage, spontaneous drug reporting systems (SRS) come to play. These systems empower physician with the tools to report suspicions on certain drugs to a pharmacovigilance centre. Latest advances take these tools even further, completing the drug loop by providing a complete reporting infrastructure to pharmacists, clinicians and patients. Pharmacovigilance centres task is to collect these reports, generating enough data to inform stakeholders of potential risks as soon as they appear in the system. Despite the invaluable data coming from SRS, their data alone are meaningless in most scenarios. Viewing SRS as independent entities makes it nigh impossible to establish direct relationships between the causes (a drug, or drug interaction) and consequences (a phenotype). Hence, to extract meaningful insights from these SRS records, one needs to rely on advanced data mining techniques. These will provide distinct perspectives over acquired data and their connections to other information topics.

Another strategy was put in place to complement spontaneous reporting systems. Intensive monitoring systems rely on prescription data, forcing drug prescribers to ask about any adverse reaction during the drug intake cycle. Once these data are collected, they are processed for signal evaluation. Unlike SRS, which is based on monitoring specific drugs over a controlled time period, intensive reporting relies on a non-interventional observational cohort. Hence, generated data are much nearer real-world scenarios than data provided obtained through SRS. Intensive reporting also renewed the interest in and importance of health information systems and general practice research databases.

Modern regional and national health information systems tend to store miscellaneous information regarding patients' clinical history, including drug prescriptions, vaccinations, height, weight or laboratory test results, among others. These wide collections of data are traditionally a good general representation of region demographics. Furthermore, collected data are already used for pharmacoepidemiology, disease epidemiology and, to a lesser extent, drug usage or pharmacoeconomics [158, 159]. From a pharmacovigilance perspective and in a European or worldwide scale, the amount and type of data collected in these databases is of tremendous importance for an improved post-market drug evaluation.

Despite the myriad of international developments in these fronts, most efforts approach this problem from a pre-market approach, focusing on conditioning drug approval and defining guidelines for risk management plans. Hence, modern projects such as the EU-ADR project, define a proactive strategy for post-marketing drug assessment. The foundation for this strategy is doing an in-depth data mining of the wealth of electronic health records to generate filtered data that can be easily substantiated through distributed computational tools. The final output, a ranked signal list, provides a broad look over identified signals and their significance in health risk.

4.1.2 The European EU-ADR Project

The European EU-ADR Project exploits partner data from national electronic healthcare records (EHR) and health information systems (HIS) of about 20 million European patients, channelling it through state-of-the-art distributed computing software and enriching signal detection. This large-scale drug safety monitoring relies in various mining, epidemiological, statistical and computing techniques to assess acquired data and generate a ranked signal list (Figure 4-1).

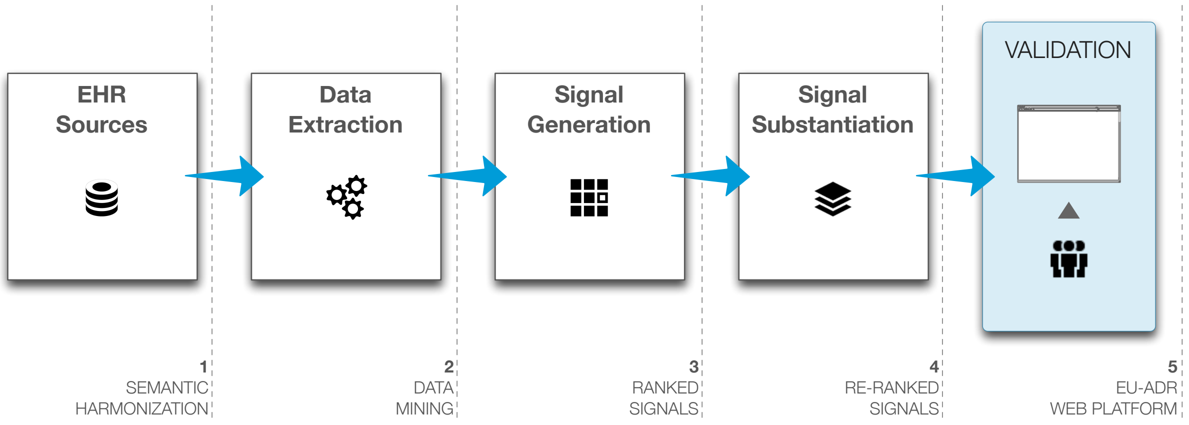


Figure 4-1. EU-ADR data flow. 1) Semantic harmonization methods prepare data from millions of records originated from distributed electronic health records and general practitioners databases. 2) Data are extracted from harmonized resources and mined for signals. 3) An initial signal evaluation generated a ranked signal list. 4) Signals are re-ranked after further processing in workflows and evidence combination services. 5) The EU-ADR Web Platform delivers the results to pharmacovigilance researchers for the final system validation.

This innovative approach for the early detection of adverse drug reactions is available to end-users through an online drug assessment tool: the EU-ADR Web Platform. In this tool, project web services and workflows are composed to test drug-event data against known literature or related protein-drug interactions, for example. With anonymous access to EU-ADR data, any user can evaluate any represented drug within the entire project scope, making this platform a unique tool for advanced drug studies.

4.1.3 Drug Safety Signal Substantiation

To coordinate efforts at a European level, EU-ADR project partners decided to distribute data extraction, signal filtering, signal substantiation, and evidence combination tasks through independent web services. These services are then composed using Taverna workflows, and the standardized input/output can be analysed in customizable graph visualization tools, such as Cytoscape [160].

Adoption of Workflow-based approaches

Within the EU-ADR context, the search for meaningful relationships amongst drugs and clinical adverse reactions must be supported by the publicly available knowledge regarding the involved drugs and phenotypes. In this pharmacovigilance effort, a better understanding of drug-event pairs is obtained through two approaches. On the one hand, the signal filtering process searches for drug-event association previously reported in biomedical literature and biomedical knowledge bases. On the other hand, the signal substantiation process looks for signals' causal inferences. That is, drug and event protein profiles are explored for common ancestry, which establishes indirect relationships between drugs, events and proteins.

In general, these data exploration tasks could be accomplished in a single informatics infrastructure. However, due to the distributed nature of the EU-ADR project, the consortium opted for building web services. These web services must be interoperable within the EU-ADR partnership, but also independent. With the former, services must obey to strict software interoperability guidelines amongst the various EU-ADR software modules. Nevertheless, with the latter, services should be designed in a way that enables its standalone use, beyond the EU-ADR scenario.

With this distributed service architecture, the EU-ADR data analysis framework requires the adoption of advanced service composition techniques to attain the expected results. Hence, the adoptions of workflow-based approaches to integrate each independent module in a high-level analysis pipeline. Web services can be executed independently or combined with other services in Taverna. This required the definition of a “service communication language”, the EU-ADR schema, discussed further in this chapter.

The workflow approach also enables accessing and executing services in a very straightforward fashion, with no programming required. By wrapping each service in its own Taverna workflow, and making it available for download, anyone can use it in a local Taverna instance. With a set of well-documented and simplified workflows, EU-ADR signal filtering and substations tools are available for use in any software infrastructure, inside and outside the EU-ADR project scope.

Workflows for Automated Data Analysis

EU-ADR workflows are grouped according to two areas of work: signal substantiation and signal filtering. For signal filtering, two workflows have been developed by project partners.

The ADR-FM workflow uses the MeSH® annotations associated with Medline® literature to automate the search of publications related to given drug-event connections. By analysing the *chemically induced*, *adverse effects* and *pharmacological action* subheadings, this method determines if the adverse drug reaction (or a similar one) has been previously published. One should not ignore the caveats of this automated mechanism. In spite of the algorithm theoretical quality, expert reading of highlighted articles is advisable. Therefore, in the EU-ADR framework, only signals matching at least 3 published elements are considered risky.

The ADR-FD workflow explores associations between drugs and phenotypes that have been previously reported in the literature (Medline®) or in drug databases (DailyMed® or DrugBank). The algorithm behind this service identifies drug and side effects co-occurrences in an indexed data warehouse, customized for EU-ADR’s information demands.

At last, the ADR-SS workflow performs the signal substantiation generating the protein interaction graphs from the drug-event pairing. This involves searching for proteins targeted by the drug and associated with the clinical event, and for biological pathways. The algorithm generates Drug-Target and Event-Protein profiles that are searched for common sets of proteins, the intersecting portion of the graph.

These three workflows accept a similar input, a drug-event pair, and produce a similar output, standardized XML. This way they can be easily integrated in a single applications, fulfilling the EU-ADR project initial goals. The EU-ADR Web Platform enacts these workflows, using custom data inputs and displaying custom data views over output data.

4.1.4 Requirements Analysis and Design Issues

From the deep evaluation of pharmacovigilance state-of-the-art and the interactions with various EU-ADR project partners, we uncovered a set of high-level requirements, which drove the development of a new pharmacovigilance strategy and resulted in the creation of the EU-ADR Web Platform. These general requirements are the following: (R1) support for complex pharmacovigilance workflows, (R2) data mining results integration, (R3) data sharing, (R4) signal substantiation, (R5) availability and (R6) exchange with software tools. The requirements are detailed next:

- → **(R1) Support for complex pharmacovigilance workflows.** The EU-ADR Web Platform must support complex pharmacovigilance studies, which require interactions amongst multiple players.
  - – **(R1.1) Service/workflow interoperability.** Software interoperability must be assured for workflows and services within and beyond the EU-ADR project.
  - – **(R1.2) Taverna workflow integration and execution.** The EU-ADR Web Platform must allow the straightforward integration and execution of the project partner's workflows for data analysis.
- → **(R2) Data mining results integration.** With the EU-ADR project collecting harmonized data from millions of patients, new methods must be developed to enable the correct display and assessment of acquired data.
  - – **(R2.1) Suitable treatment of data files.** The majority of the EU-ADR Web Platform's users will submit their private data in Excel or CSV files.
- → **(R3) Data sharing.** Adequate data sharing features are essential for performing successful research endeavours in any context. Hence, the EU-ADR Web Platform must include data sharing functionalities to enable the reproducibility and validation of results for any selected researcher.
  - – **(R3.1) Promote collaborative research.** Users must be able to share submitted data with selected associates or within the context of a larger user group.
  - – **(R3.2) Research reproducibility.** Collected data must be constantly available for analysis, substantiation and exploration.

- **(R4) Signal substantiation.** The substantiation of submitted signals, which includes the processing of these signals in all the project's workflows, must be facilitated and the analysis of results streamlined.
  - **(R1.1) Signal risk evaluation.** New interfaces must be setup to display signal risk evaluation results from the multiple workflow executions, enabling a clear view regarding the importance of a relationship between a drug and an adverse event.
  - **(R4.1) Combined risk assessment.** EU-ADR workflow results must be combined, using an independent evidence combination algorithm, into a summary risk classification of high risk (H), medium risk (M) or low risk (L).
- **(R5) Availability.** The EU-ADR Web Platform must be publicly available at all times for all registered users.
  - **(R5.1) Highly interactive web-based workspace.** A web-based workspace must be implemented to support new analysis, the visualization of results and the sharing of data. Highly interactive interfaces should be used to facilitate access to the wealth of EU-ADR data.
- **(R6) Exchange with software tools.** In addition to (R5), external data evaluations must also be supported.
  - **(R6.1) Export results.** Exporting direct results from Taverna executions should be possible.

These requirements match the challenges brought about by various authors in the pharmacovigilance research field over the last few years, as highlighted in Table 4-1.

Table 4-1. Match between published literature challenges and the requirements devised for the EU-ADR Web Platform development.

	R1	R2	R3	R4	R5	R6
Meyboom *et al.* [154]	●			●		
Wadman *et al.* [155]	●					
Bauer-Mehren *et al.* [156]	●			●	●	●
Harmark *et al.* [157]			●		●	
Wood *et al.* [158]	●	●				
Shannon *et al.* [160]						●

	R1	R2	R3	R4	R5	R6
Coloma *et al.* [161]	●	●	●			
Trifirò *et al.* [162]	●	●	●	●	●	

4.2 The EU-ADR Web Platform

Workflow enactment within the EU-ADR project is essential to combine the variety of deployed web services in a single integrative data analysis pipeline. To tackle this challenge, the workflows were designed using the Taverna workbench. This approach permits both the local standalone execution of workflows, using Taverna, as well as the inclusion of workflows in existing applications, through Taverna's command line interface, which required the development of a new workflow execution engine. With a standard service data exchange language and a customized workflow execution tool, all pieces were in place to build the EU-ADR Web Platform.

4.2.1 Exploring Service Composition for Interoperability

Computational biology evolution has also greatly improved *in silico* experimentation and, consequently, research reproducibility. This is fostered by the growing number of organizations offering some kind of programmatic access to their knowledge bases and applications. Web services offer straightforward, published, application programming interfaces for interaction with and within other systems.

From a bioinformatics perspective, the tasks for analyzing and exploring *in silico* experiments data are traditionally linked in a way that can be easily mapped to a software workflow. By creating autonomous data flows between multiple services, the use of scientific workflows greatly improves the clinicians and researchers computational tasks. In section 2.2 we detailed a large number of data sources, services and applications, most of which provide services that can be easily composed into new meta-applications.

DynamicFlow

With the Internet gaining momentum as a development platform, we are assisting a shift in the computational paradigm: moving from desktop applications to web and mobile. Therefore, we designed DynamicFlow (Figure 4-2), a solution to promote autonomous dynamic service composition in a web-based environment, requiring nothing else than an Internet browser [8, 9].

DynamicFlow's goal is to offer access to a collection of visual components that bind public web service as wrappers described following a predefined ontology. This description contains essential information about the service execution. The content of each service may vary but it has to follow a set of minimal mandatory elements:

- **DisplayName**, defines the name that will appear in the component list;
- **Description**, small description and relevant information that will be shown within the Help section;
- **Input** and **Output**, the **Type** and the **Value** of the service inputs and outputs; these elements are essential in the workflow execution process. Each workflow is validated by checking the consistency of each component inputs against the previous component outputs;
- **Specie**, list of the species where this component can be used;
- **XmlString**, XPath query used to select the correct objects from each service output.

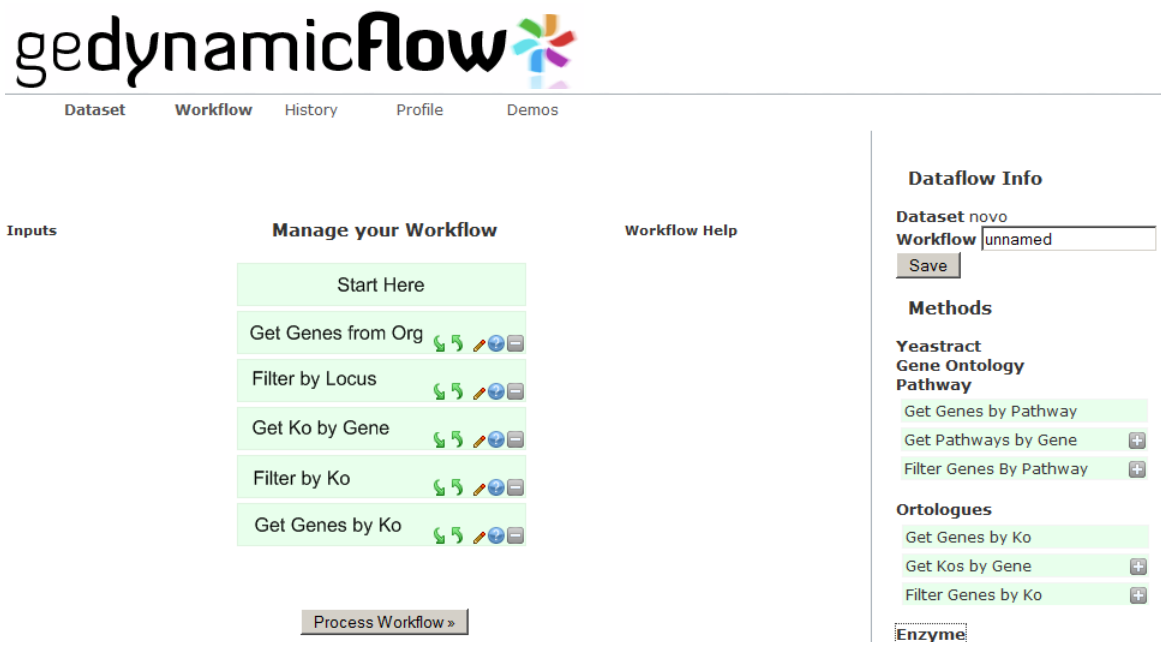


Figure 4-2. DynamicFlow workspace interface. Central area for workflow edition and right sidebar with available task listing.

These services semantics are a primitive version of later developments described in WAVe and COEUS. Despite this, the principle behind its use is the same: provide a content description that enhances information queries and promotes automated interactions among distinct resources.

Interoperability research in DynamicFlow platform was focused around three key areas: the web-oriented architecture dividing processing workload between the web server and the browser; the semantic description of web services and the agile web interface for workflow composition.

First, client-side processing enables the creation of new features and increases application performance by reducing client-server data exchanges. With more advanced client-side development frameworks appearing daily, web clients' data handling capabilities are evolving, enabling intensive data processing tasks on the browser.

Second, we designed an ontology with comprehensive semantics to describe the services that are part of the workflow. In addition to making the framework generic and enhancing the application flexibility, it was an initial endeavour on the description of remote services for data exchanges.

At last, continuing with the modern client-side capabilities, the interface relies completely on the browser, does not require any special plug-in, and adopts a traditional desktop metaphor to create a richer environment for the design and execution of workflows.

Taverna

The main goal of workflow management applications is to abstract the programming side of the application, enabling the creation of comprehensive workflows without writing a single line of code.

From the various software platforms for combining services, Taverna emerged as the *de facto* standard for desktop-based workflow management in the life sciences [163]. Taverna's success is mostly due to its flexibility and variety of available *processors*. In this context, a processor is an activity that can be a part of workflow (Figure 4-3). Taverna's processors can be web services, XML splitters, file system handling tools, string manipulation, Excel spreadsheet readers or custom Java code, among others. These allow researchers to create complex service composition environment just by dragging and dropping boxes in Taverna's workbench.

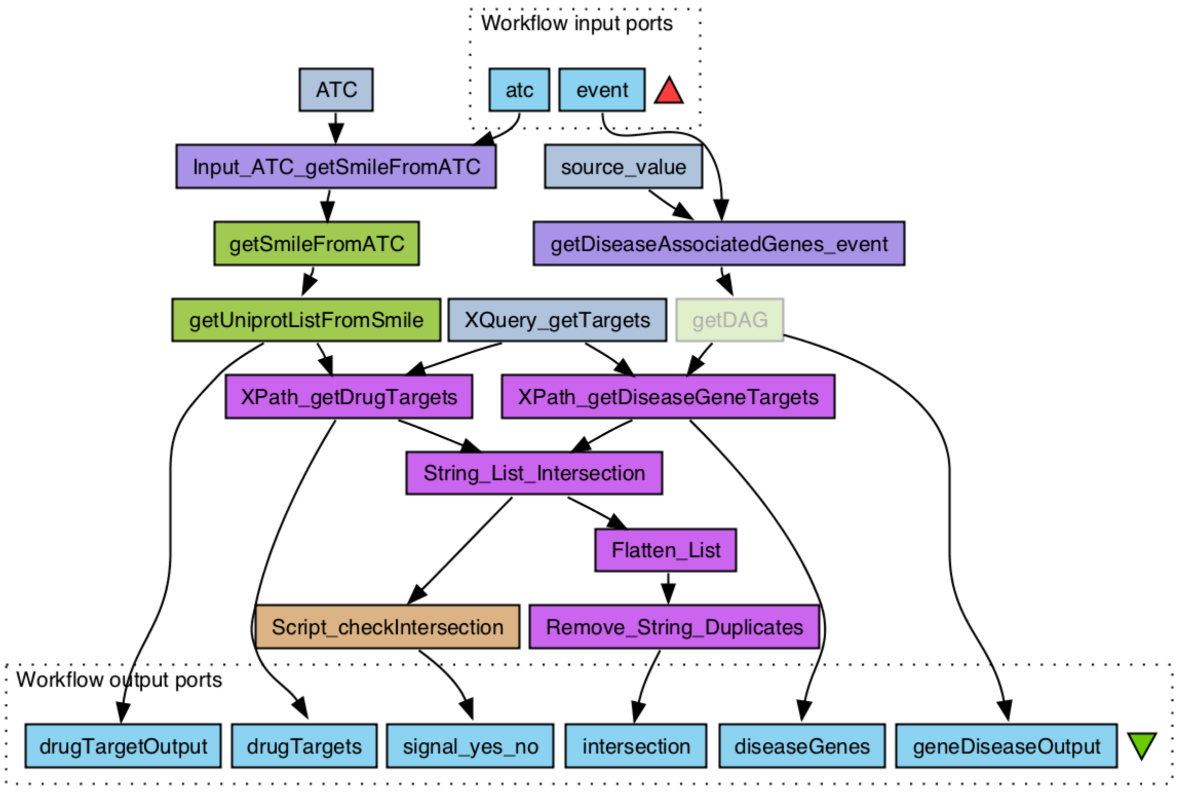


Figure 4-3. EU-ADR's signal substantiation workflow: the inputs, a drug (an ATC code) and an adverse event, are processed through multiple activities, including UniProt and SMILE services, to generate multiple outputs, including drug targets, protein interaction networks and genes.

4.2.2 Application Setup

The EU-ADR Web Platform is sustained by a distributed computerized system combining multiple components in a single software ecosystem. The platform can be logically divided in five key areas: the client application, the application engine, the workflow execution engine, the evidence combination engine, and the external workflows – Figure 4-4.

As mentioned before, the EU-ADR project has deployed 3 workflows for drug-event pair data exploration and assessment. These workflows play an active role in the EU-ADR Web Platform, as they are required for the data analysis features. The challenging tasks of accessing and executing these Taverna workflows required the development of a new workflow execution engine, enabling real-time web-based communication with Taverna workflows.

The server software includes the workflow execution and evidence combination engines, the platform database and the application engine. The latter is the main EU-ADR Web Platform controller, coordinating all involved components. Using the Model-View-Presenter architecture, we rely on Google Web Toolkit[15] (GWT) as application engine.

[15] https://developers.google.com/web-toolkit/

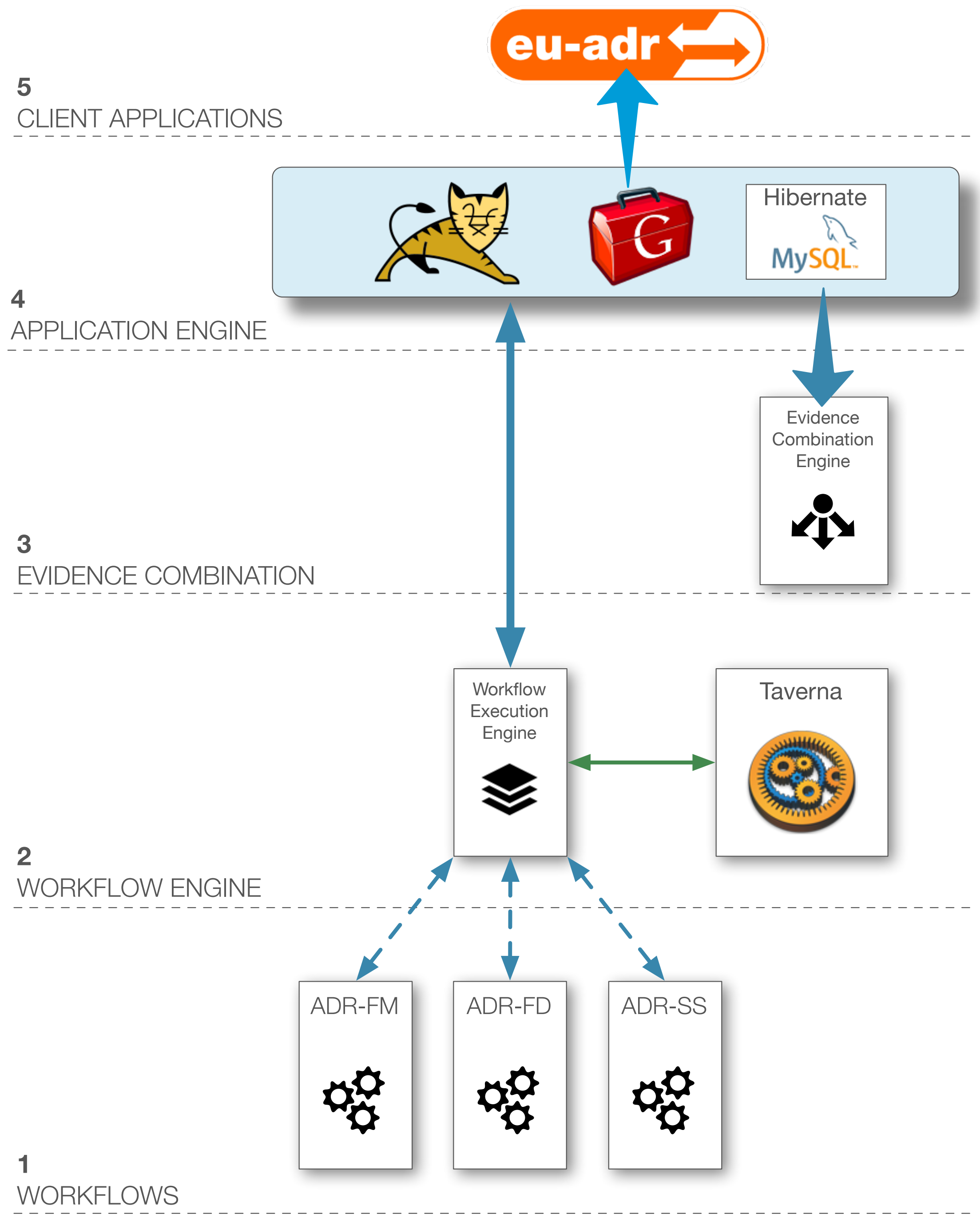


Figure 4-4. EU-ADR Web Platform architecture. 1) External distributed workflows are integrated with the workflow engine. 2) The created workflow execution engine interacts with Taverna, launching new workflow runs and processing results. 3) The evidence combination engine processes the signal list, once all workflows have been executed, to generate a final ranked list. 4) The application engine, developed with Google Web Toolkit and served trough Apache Tomcat, controls the entire application execution, from triggering workflow execution to data persistence in the MySQL database. 5) The EU-ADR Web Platform online application is a unique entry point to advanced pharmacovigilance features.

The client application uses a myriad of advanced user interaction components to provide a clean perspective over the huge drug datasets and easy access to data exploration features. The EU-ADR Web Platform server-side is controlled by GWT and implemented in

Java, with Apache Tomcat serving the application. Data are stored in a MySQL database, assuring the persistence of the application business model.

For an improved data handling, Hibernate[16] was used for the data abstraction layer and object/relational mappings, thus reducing undesirable database coupling with the application. This shields the development from future changes in the domain model storage system and eases the use within the Java object-oriented environment.

Miscellaneous additional components were also used, such as Spring Security for improved security features, Apache POI[17] for enhanced data import and export, Google Guice[18] for dependency injection, Log4j for logging purposes and Apache Maven[19] for an enhanced project dependency management, building and deployment.

To improve on GWT's user interactions library, we adopted the richer Ext GWT package (GXT)[20]. This extends widgets bundled with GWT core distribution to provide a more complete set of user interaction features required by the Web Platform's interface.

The combination of GWT's basic widgets with GXT ones was further improved with Google Gin[21] for dependency injection, achieving a truly decoupled architecture.

Investigation of any drug-event pair does not end after the primary workflow relative risk assessment, as evidence needs to be combined to reach a final score to help separate spurious signals from potential adverse drug reactions.

To perform these new combinations, the evidence combination engine uses the Dempster-Shafer theory to reach a conclusion about the belief that a drug-event pair is potentially an adverse reaction [164]. This combination process takes in account the results of all workflows and any additional relevant data provided in the drug-event dataset.

4.2.3 Data Exchanges

The EU-ADR project web services required setting up a common data description and exchange language. All workflows and web services had the same input data types and it was adamant that the service outputs should be as similar as possible. Consequently, the need for this "communication language" pushed forward the study and evaluation of miscellaneous standardization technologies. Since we were dealing with distributed

[16] http://www.hibernate.org/
[17] http://poi.apache.org/
[18] http://code.google.com/p/google-guice/
[19] http://maven.apache.org/
[20] http://www.sencha.com/products/gxt
[21] http://code.google.com/p/google-gin/

services and were looking for the best way to exchange data in a format that could be easily read and parsed in any programming language, we opted for modelling a custom XML schema using XSD.

The EU-ADR schema is divided in two files. The first, **Common Types**[22], describes general data types used within the project and can be easily adapted for scenarios beyond EU-ADR. The second, **EU-ADR Types**[23], defines the structure for signal filtering and substantiation data exchanges within the EU-ADR project context. These schemas allow for a smooth integration of the different modules in Taverna workflows and in other application development environments. Furthermore, using XSD enables both content and structure validation for EU-ADR services' input and output data.

Figure 4-5 displays the overall **EU-ADR Types** structure. The document is rooted in **relationships**, which possess **globalScore** and **relationships** elements. A relationship is a drug-event combination, using ATC codes for drugs, **sourceId**, and internal event identifiers, **targetId**. Global score represents the mean score for all relationships in the input/output data flow, detailed individually in the **partialScore** element. Additional service metadata are included in the **creator**, **observationDateTime** and **informationSources** elements. The latter is further divided in multiple elements, comprising the justification for the relationship score. These include database identifiers, relationship types, discovered interactions and evidence types, among others.

With these schemas available online, web service developers are able to formally represent their services' input and output in a way that every partner understands and enabling the required service-to-service and workflow-to-workflow interactions.

[22] http://bioinformatics.ua.pt/euadr/common_types.xsd
[23] http://bioinformatics.ua.pt/euadr/euadr_types.xsd

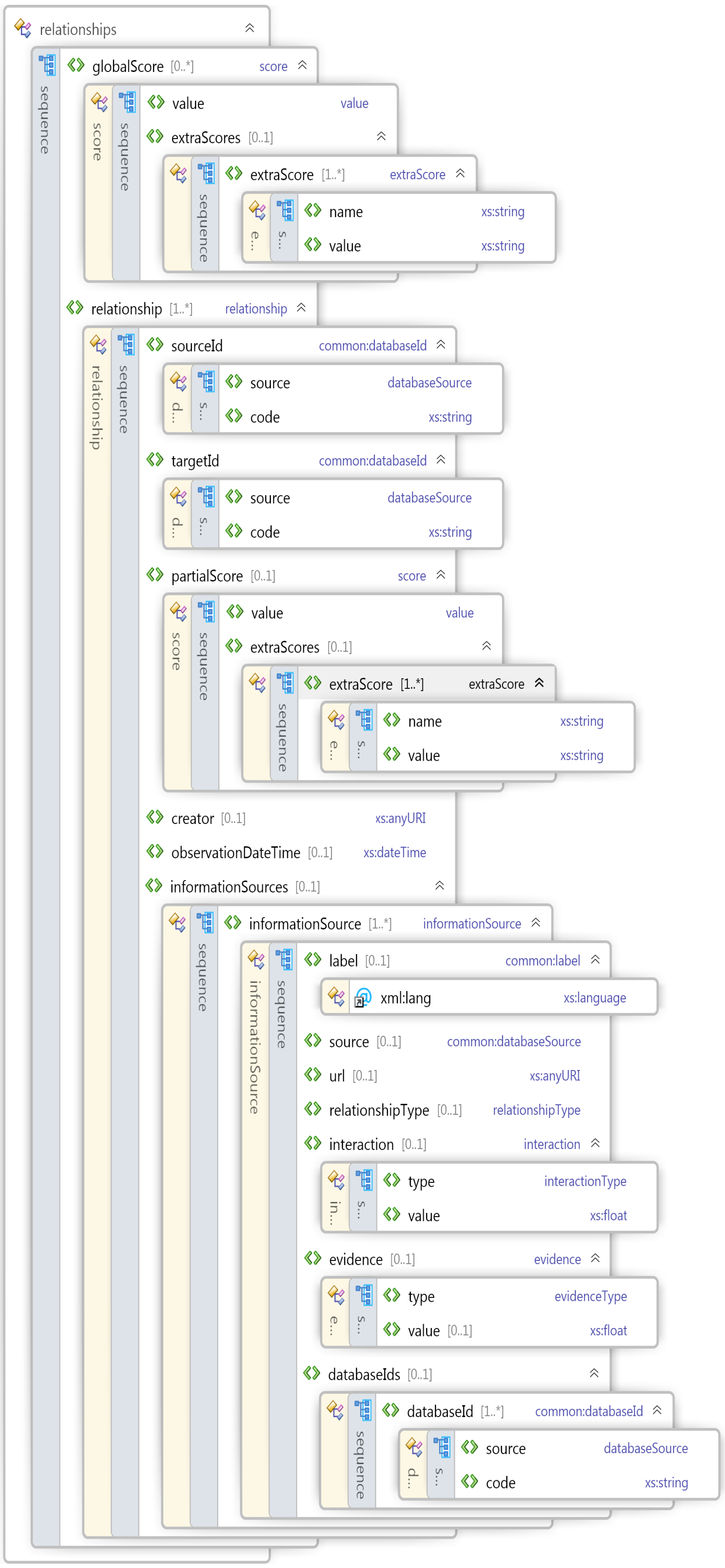


Figure 4-5. Relationship diagram overview for EU-ADR's interoperability schema.

4.2.4 Wrapping Taverna Workflows

Integrating EU-ADR workflows within the Web Platform was not a trivial task. Taverna 2 workflows are stored in a XML file with the SCUFL 1.2 specification schema, making the read/parse tasks very cumbersome to implement. Moreover, we need to feed the services with input data, manipulate intermediate results and extract the resulting output documents.

To perform these tasks, a new workflow execution engine was developed. This Java tool is included within EU-ADR Web Platform's code and enables the execution of Taverna's command line interface with custom input arguments. These parameterized system calls run in their own independent OS process, increasing the overall platform performance and scalability. Workflow executions are also a background non-blocking asynchronous process. For EU-ADR Web Platform users, this means that they can use the application and their data whilst workflows are running in the background.

Figure 4-6 illustrates the steps required to execute the workflow in the web platform server. From user input to system output the platform executes the following actions in order:

1. Signal substantiation for one or more drug-event pairs is triggered by users in the Web Platform interface.
2. An XML file with the relationship set is generated and its path supplied to the workflow execution engine along with the selected workflows path. The workflow is then launched by a system call.
3. The EU-ADR Web Platform internal workflow reads and translates the XML input, invoking the remote web service with the transformed input.
4. The external web service processes data and sends it back (XML) to Taverna's command line interface.
5. Additional intermediate processing takes place within the workflow execution engine before the following service is contacted.
6. The service processes data and replies with its XML output.
7. The workflow execution engine assembles all retrieved data into one or more XML files: the final output.
8. Taverna's command line interface finalizes the system call and the web platform scans the provided output directory for the workflow output XML files.

9. Workflow results are parsed and stored in the web platform's database and the XML files moved to a local repository for permanent storage and future reference.
10. The browser interface is updated with the substantiation results, displaying new information views with collected evidence.

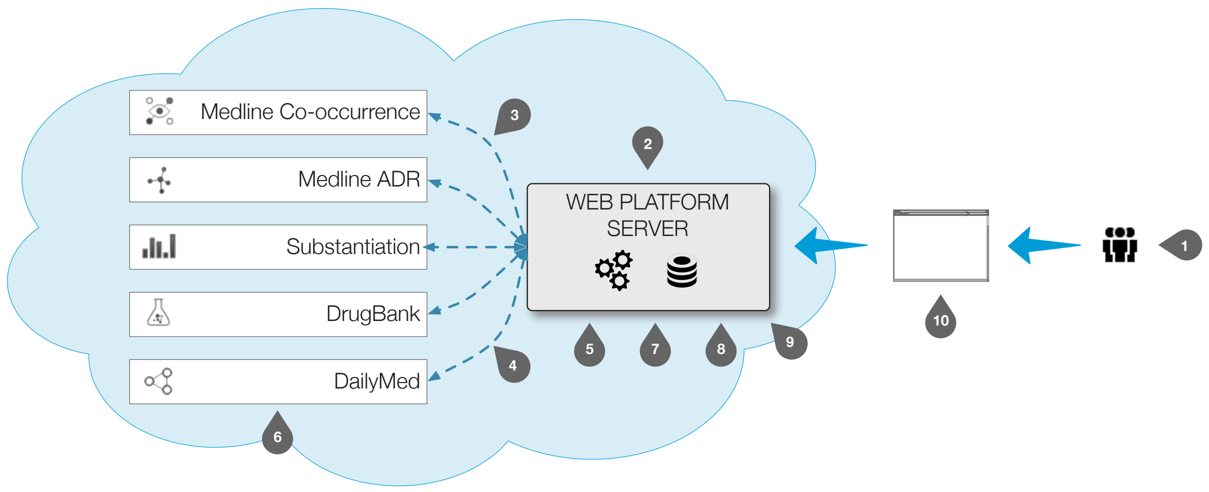


Figure 4-6. EU-ADR Web Platform workflow iterative execution process.

Once data have passed through the workflow execution engine, it is promptly available for user assessment and for further statistical analysis. The evidence combination algorithm, which is outside this document's scope, performs a Dempster-Shafer analysis to improve the final score precision and help separate spurious signals from potential adverse drug reactions.

4.2.5 Features and Usability

Service-to-Service and Workflow-to-Workflow Data Exchanges

Fully automated web service interoperability requires the creation of a common data exchange language, allowing for environment-independent machine-to-machine communication. This basic need, discussed with detail in section 3.2, was a top priority within the EU-ADR project. Moreover, considering the distributed nature of modern software and common large-scale consortiums promoting research, one can easily identify this requirement as an overarching demand. The developed schema set, divided in common and project-specific data types, leveraged a faster development of new services within the EU-ADR project.

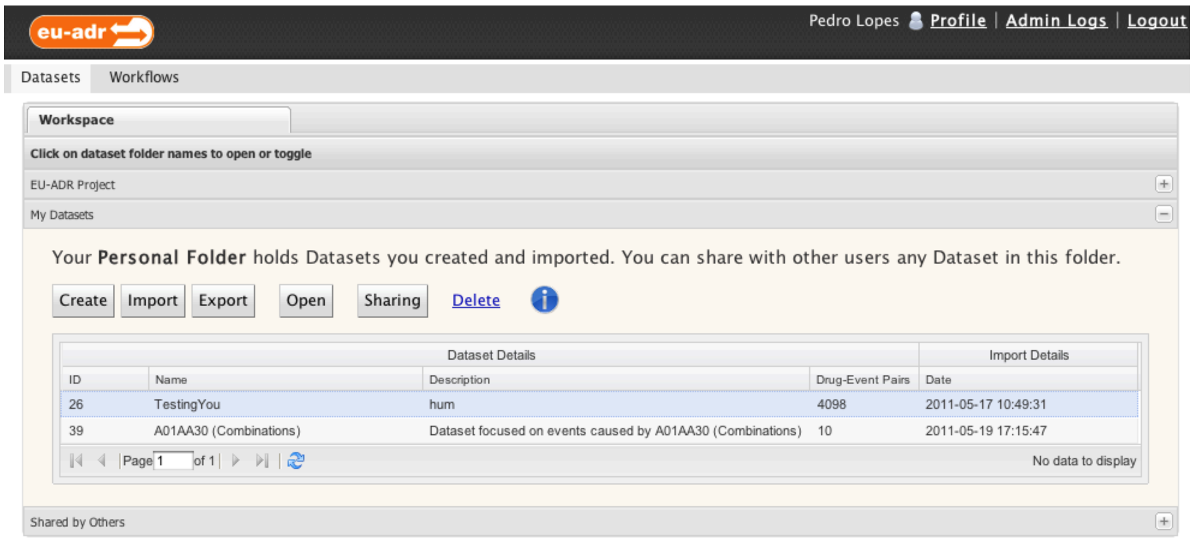
The definition of these kinds of standards is the initial step towards fully interoperable software. The usage of SOAP for service communication only solves the technological problem of exchanging data. Transferred data must also be read and understood by all software involved in the process. With the EU-ADR schema, and others alike, we can exchange data between distributed systems and easily apprehend the meaning of shared data. The true value behind advanced service composition strategies only surfaces with the adoption of data exchange standards or shared data models.

Advanced Drug Studies

EU-ADR Web Platform's key feature is the execution of advanced post-marketing adverse drug reaction studies. Registered users are able to upload and analyse drug-event datasets, create targeted drug studies and collaborate with their research peers through the available sharing features.

The invite-based registration system allows selected researchers to join the web platform by giving them access to a personal closed workspace. In this area they can browse existing datasets (personal or shared); upload custom drug-event pair datasets; or create drug-specific datasets, based on the overall web platform data. Since this application was built as a response to EU-ADR's project needs, the system is pre-filled with data mined from over 30 million European patient records. Whilst these data are not directly available, they can be used to evaluate specific drug adverse reactions on the project workflows.

Next, EU-ADR Web Platform's features are highlighted in a traditional custom drug study scenario. The initial workspace (Figure 4-7) offers access to each user's datasets, organized in two distinct sections.



Your Personal Folder holds Datasets you created and imported. You can share with other users any Dataset in this folder.

Dataset Details				Import Details
ID	Name	Description	Drug-Event Pairs	Date
26	TestingYou	hum	4098	2011-05-17 10:49:31
39	A01AA30 (Combinations)	Dataset focused on events caused by A01AA30 (Combinations)	10	2011-05-19 17:15:47

Figure 4-7. EU-ADR Web Platform view for personal dataset listings.

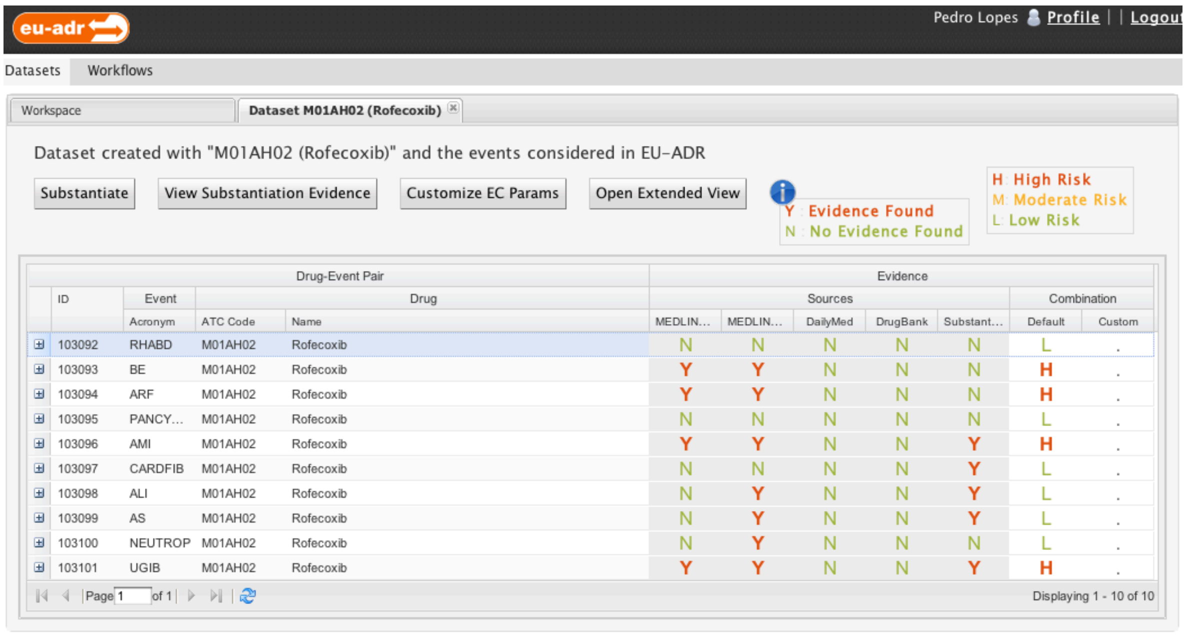
As the name suggests, **My Datasets** contains the datasets created by the user and **Shared by Others** lists datasets shared between users. For users identified as a part of the EU-ADR project, a third section with project datasets appears on top. Within this view, users can access various dataset features: the **Create**, **Import**, **Export**, **Open**, **Sharing** or **Delete** options are enabled for all datasets in the **My Datasets** tab. For testing purpose, we create a new dataset targeting a specific drug.

Following the reports on **rofecoxib** and its withdrawal from the market, we test this drug to validate the system. This drug for treating osteoarthritis, acute pain conditions, and dysmenorrhoea, is well known for causing severe cardiac problems. Using the **Create** button we just need to start typing the drug name and the type-ahead system will provide the matching drugs with respective ATC code.

Once we select the drug, the system generates a new dataset matching the selected drug with all EU-ADR events in the system, as shown in Figure 4-8. Then, we move to the **substantiation** of all signals. During this process, the multiple drug-event combinations will be provided to EU-ADR workflows, which will analyse them individually and provide evidence to support the drug risk or to mark the pair as a non-valid signal. The drug-event pair, a **relationship**, will traverse the 5 workflows and the evidence combination service, generating data to help identify **rofecoxib** risk in multiple events. These data are visually marked as **red Y** or **green N** for workflows, when data for adverse reaction was found or not, respectively. The evidence combination service performs the final statistical analysis, classifying each pair risk as high (**H**), medium (**M**) or low (**L**).

Once the processing finishes, users can explore evidences for each pair individually. In this pane (Figure 4-9), the wealth of data provided by each workflow can be easily exploited. Furthermore, evidence data includes connections to external applications supporting the evidence, including UniProt proteins and Medline literature.

Once the processing finishes, users can explore evidences for each pair individually. In this pane (Figure 4-9), the wealth of data provided by each workflow can be easily exploited. Furthermore, evidence data includes connections to external applications supporting the evidence, including UniProt proteins and Medline literature.

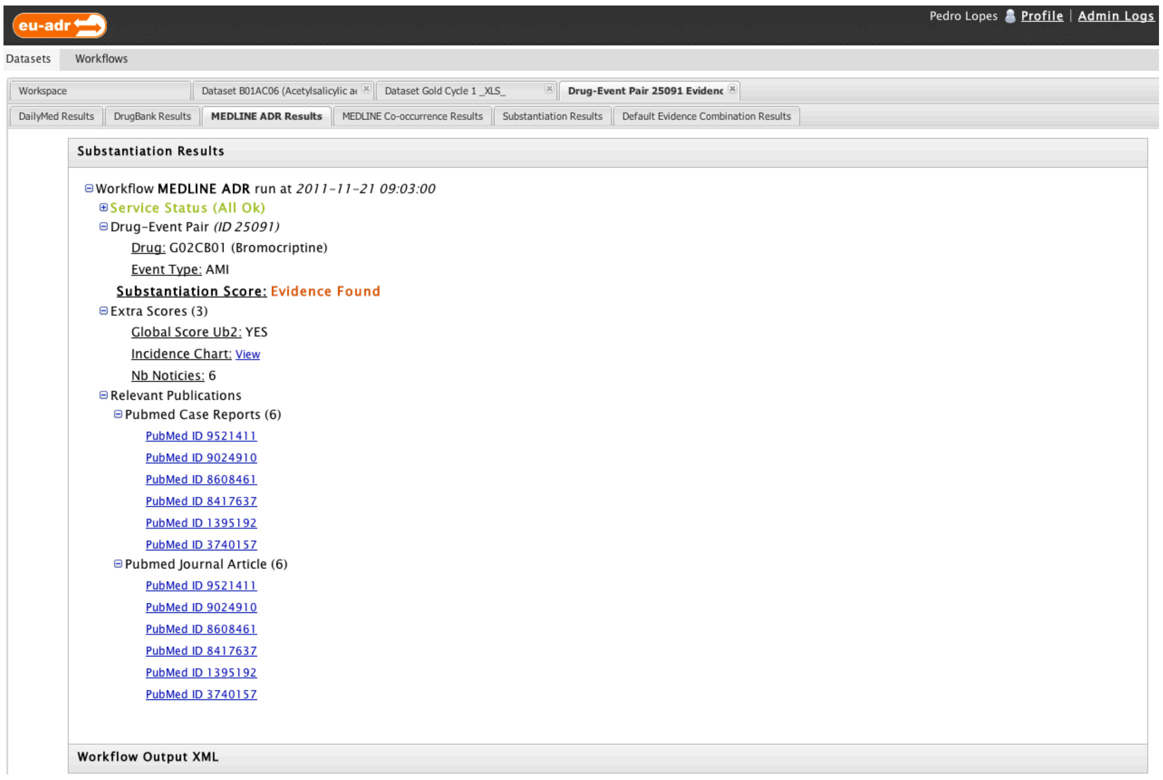


Dataset created with "M01AH02 (Rofecoxib)" and the events considered in EU-ADR

Drug-Event Pair				Evidence						
ID	Event	Drug		Sources					Combination	
	Acronym	ATC Code	Name	MEDLIN...	MEDLIN...	DailyMed	DrugBank	Substant...	Default	Custom
103092	RHABD	M01AH02	Rofecoxib	N	N	N	N	N	L	.
103093	BE	M01AH02	Rofecoxib	Y	Y	N	N	N	H	.
103094	ARF	M01AH02	Rofecoxib	Y	Y	N	N	N	H	.
103095	PANCY...	M01AH02	Rofecoxib	N	N	N	N	N	L	.
103096	AMI	M01AH02	Rofecoxib	Y	Y	N	N	Y	H	.
103097	CARDFIB	M01AH02	Rofecoxib	N	N	N	N	Y	L	.
103098	ALI	M01AH02	Rofecoxib	N	Y	N	N	Y	L	.
103099	AS	M01AH02	Rofecoxib	N	Y	N	N	Y	L	.
103100	NEUTROP	M01AH02	Rofecoxib	N	Y	N	N	N	L	.
103101	UGIB	M01AH02	Rofecoxib	Y	Y	N	N	Y	H	.

Page 1 of 1 Displaying 1 - 10 of 10

Figure 4-8. EU-ADR Web Platform view for a personal dataset created for the Rofecoxib **drug (ATC code M01AH02).**


eu-adr
Pedro Lopes Profile | Admin Logs
Datasets
Workflows
Workspace
Dataset B01AC06 (Acetylsalicylic a
Dataset Gold Cycle 1 _XLS_
Drug-Event Pair 25091 Evidenc
DailyMed Results
DrugBank Results
MEDLINE ADR Results
MEDLINE Co-occurrence Results
Substantiation Results
Default Evidence Combination Results

Substantiation Results

Workflow **MEDLINE ADR** run at *2011-11-21 09:03:00*
- Service Status (All Ok)
- Drug-Event Pair *(ID 25091)*
  - Drug: G02CB01 (Bromocriptine)
  - Event Type: AMI
  - **Substantiation Score:** Evidence Found
- Extra Scores (3)
  - Global Score Ub2: YES
  - Incidence Chart: View
  - Nb Noticies: 6
- Relevant Publications
  - Pubmed Case Reports (6)
    - PubMed ID 9521411
    - PubMed ID 9024910
    - PubMed ID 8608461
    - PubMed ID 8417637
    - PubMed ID 1395192
    - PubMed ID 3740157
  - Pubmed Journal Article (6)
    - PubMed ID 9521411
    - PubMed ID 9024910
    - PubMed ID 8608461
    - PubMed ID 8417637
    - PubMed ID 1395192
    - PubMed ID 3740157

Workflow Output XML

Figure 4-9. EU-ADR Web Platform view for signal substantiation results for the signal combining the Bromocriptine **drug (ATC: G02CB01) with the Acute Myocardial Infarction (AMI) group of adverse drug reactions. The list of relevant publication from the Medline ADR workflow is the highlight of this particular interface.**

4.3 Discussion

4.3.1 Service Composition for Interoperability in Bioinformatics

As discussed in this chapter, the use of service composition strategies for interoperability is gaining relevance amidst the bioinformatics community. With an ever-growing number of services, from data mapping services to semantic data enrichers, there is untapped potential for exploring service-oriented architectures in bioinformatics. Mashups and workflows built around the wealth of existing services allow developers to quickly connect data and features, without relying on local computational power.

Nowadays, planning and implementing service-based workflows are integral parts of the bioinformatics software development process. This way, entire applications simply wrap a set of composed services behind an advanced user interface. Workflow managers have also evolved, originating new tools to quickly manage, develop, update or execute complex service-based workflows. Taverna is the pinnacle of this evolution, providing the set of tools to create almost any kind of dynamic workflow in a visual-oriented fashion.

Despite its quality, Taverna is limited to desktop-based use and web-based solutions, such as DynamicFlow, are far away from the capabilities of a desktop workbench. Whilst this is not a problem for traditional lab users, developers are faced with these challenges in projects where using a workflow management tool is essential. This is the case for the EU-ADR Web Platform: Taverna workflows enable cross-project service composition for interoperability and these workflows can now be executed locally or online by any researcher.

4.3.2 Fostering Pharmacovigilance Innovation through Service Composition

The EU-ADR project embraces sophisticated pharmacovigilance research methods in an online platform providing advanced drug data exploration and assessment features. Whereas in the past post-marketing drug assessment required intense validation tasks, the *in silico* pharmacology community is now endowed with the tools to quickly analyse specific adverse drug reactions, further improving drug safety monitoring.

The computational strategies created to fulfil the initial set of requirements originated a new platform for delivering advanced pharmacovigilance studies. The EU-ADR Web Platform enables streamlined access to drug dataset analysis features, including the evaluation of results from EU-ADR workflows and the sharing of data amongst *ad hoc* research partners. All this is possible due to an architecture sustained by four outstanding innovations, highlighted next.

- The project-wide **interoperability standard** enables automated data exchanges amongst the various partners' web services. This new schema standardizes the services' input and output, making the creation of complex EU-ADR workflows possible.
- The set of **EU-ADR workflows** comprises interactions between heterogeneous services provided by distributed partners throughout Europe. With each service focusing on a particular data evaluation, the drug-event pair analysis is distributed through the service set and combined in the EU-ADR Web Platform.
- The new **workflow execution engine** provides a streamlined way to include Taverna workflows within Java applications. This makes the *ad hoc* execution of workflows in the EU-ADR Web Platform possible, allowing real-time drug-event pair data processing.
- At last, the GWT-powered **web-based workspace** makes it very easy to create custom drug-event evaluations, upload big datasets, substantiate data on-demand and evaluate the relative risk for drug-event associations crossed against a background with data mined from millions of electronic health records, publications and drug-protein interactions.

The EU-ADR Web Platform is available online, at http://bioinformatics.ua.pt/euadr/.

5. WAVE: BUILDING AN INTEGRATIVE KNOWLEDGE BASE

"Quality in a service or product is not what you put into it. It is what the client or customer gets out of it."
- Peter F. Drucker

Modern computer science technologies are essential elements to handle the growing volume of biomedical and biological data being generated everywhere, from research labs to clinical centres. Traditionally, researchers require very specific data analysis and transformation skills, ranging from the interrogation of data sources to the management and reorganization of information so it is available for input to distinct software. With more integrative tools we can enable the next generation of bioinformatics software, sustained by advanced service composition for data integration strategies.

This chapter encloses our developments towards the creation of a unique human variome portal, within the European GEN2PHEN Project context. The WAVe platform is introduced as our initial concerted effort towards solving a pertinent bioinformatics challenge with service composition strategies [11]. Collecting human variome information is essential to grasp the meaning of our genetic sequence variations and their effects. In WAVe, service composition techniques are used for the integration of data and to provide the collection of acquired data through a REST API. WAVe is publicly available online at http://bioinformatics.ua.pt/WAVe/.

From the combination of our initial assessment and our contributions to the field, including the previous chapter, we identified a clear need for more dynamic life sciences software. With so many common problems surfacing in miscellaneous large-scale research projects, it is essential to reuse and recombine components to streamline the software development process. These conclusions finish this chapter and open the discussion for the contributions detailed in chapter 6.

5.1 Human Variome Research

Human variome research has flourished over the last decade, triggered by the explosive growth of available genetic data emerging from the completion of the human genome sequence [15, 165]. The discovery of novel relationships between simple sequence changes and diseases is essential to underpin the future prospect of custom drug design and personalized patient care [166-168]. Furthermore, datasets are growing at such rate, and with such diversity of data, that they often demand custom software solutions. The correct description, publication, and enrichment of disease-causing gene variants, requires new approaches and expertise from genetic data integration developers [169-172]. Produced solutions often fail to take into account similar counterparts in the domain, reengineering everything from scratch and closing the system to future data exchanges. Consequently, the genetic variation research field lacks standardization in both interoperability and integration.

Locus-specific databases (LSDBs) are gene-centric, closed systems, designed for direct interaction with curators and focused on the linear tasks of storing and publishing discovered variants online. The software provides comprehensive variant details and their phenotypic consequences for one or a few genes of relevance to one or a few specific diseases. Despite a steady evolution in this field [173], including new genetic variation description standards [174, 175] and enhanced reference sequence formats [176], the lack of quality control and strict scope has hindered progress in this area: online data dumps and legacy systems without any scientific coherence are still widely used, hampering the process of accessing and understanding all available information through a single access point. Additionally, existing systems' available data and interfaces are limited to standalone gene analysis.

Despite LSDB completeness, researchers need access to miscellaneous resources and features while browsing gene variants: gene loci and variant information should be complemented with related proteins, pathways, or published literature, among others. Currently, this is not possible without a complex data analysis workflow involving interactions with various distinct applications.

To overcome current deficiencies in the genetic variation research field we devised a new application entitled Web Analysis of the Variome (WAVe), which empowers the extraction of LSDBs' true added value through their connection with both similar platforms and diverse external resources. WAVe's development was based on a new holistic approach detailed further in this article. The adopted strategy enables the setup of a platform for the integration of genetic variation datasets and the enrichment of the latter with connections to external resources and state-of-the-art user interaction features.

WAVe enables centralized access to existing LSDBs, aggregates genes and their variants, and integrates a multitude of scientifically relevant resources, without damaging the original work conducted by researchers in this domain. WAVe then publishes collected data through an API and a comprehensive and agile workspace, focused on the transparent access to the miscellaneous integrated applications and data sources in the human variome research field.

5.1.1 Integrating Human Variome Information

Available LSDBs fall into two main categories. On the one hand, there are several legacy systems, which are often just data dumps in HTML or PDF format, listing variants in an unstructured fashion. Such systems generally display variants in tables or include them in free-text summaries and, consequently, it is difficult or impossible to extract specific mutation information. On the other hand, sophisticated LSDB software packages emerge, including MUTbase [177], Universal Mutation Database (UMD) [178], and Leiden Open Variation Database (LOVD) [179, 180].

On a distinct perspective, another caveat is the political obstacle to data integration created by the reluctance in the LSDB community to share data [181]. LSDB curators have major concerns regarding data sharing, especially with respect to issues such as data quality, authorship, and ownership. Particularly, curators fear that by aggregating their data they will lose control over it: central repositories will be able to display it without proper attribution, use it for commercial purposes, or misrepresent clinical information leading to inappropriate interpretations.

Various projects have recently started to address these problems, through both the development of integrative software tools and the creation of recommendations and standards for LSDB data sharing. BioGPS is an example of the former, collecting data from gene-related resources and presenting it in a customizable workspace [182]. Despite displaying resources in a modern approach, BioGPS lacks detailed access to genetic variation datasets. Another tool in this domain is DRUMS, which provides access to variants from multiple genes gathered from a wide array of databases [183]. Although genetic variation information is very complete, DRUMS suffers from the recurring common issues in the LSDB software domain, in that the available information is deep with respect to the variome, though narrow in a holistic life sciences perspective.

This domain is also the subject for several larger initiatives such as MutaDATABASE[24], PhenCODE[25], or GEN2PHEN[26]. However, these are on-going projects whose results are still gaining traction in the LSDB research community, as each of them still requires some technology learning and development effort. Adhering to new standards will entail deep revisions in current systems: integrative data models must be changed and new interoperability features must be added, resulting in a need for architectural revisions in already stable systems. Furthermore, legacy systems that lack funding or curator interest will be lost within this necessary evolution.

5.1.2 The European GEN2PHEN Project

The "Genotype-to-Phenotype: A Holistic Solution" project (GEN2PHEN) is focused on the development of tools that will connect online life sciences resources containing information spanning from the genotype – the human genetic sequences – to the phenotype – the human visible traits such as hair colour or penchant for a specific disease. Research in this project touches miscellaneous areas like gene sequencing and expression, genotyping, SNP mapping and pharmacogenomics. These are essential for realizing the individualized healthcare premise, delivering the most effective treatment for a patient according to his clinical history, physiology and genetic profile and the molecular biology of the disease [184].

In the future, personal electronic health records (EHR) will be enriched with genetic information required for more personalized treatments. Two research fields are essential to complete EHR datasets: pharmacogenomics and genetics. Pharmacogenomics studies variability in drug response, which comprises drug absorption and disposition, drug effects, drug efficiency and adverse drug reactions [185]. Genetics profiling of diseases provides new insights on the classification and prognostic stratification of diseases based on molecular profiling originated in microarray research [186]. Both these fields will generate a tremendous amount of heterogeneous data that needs to be integrated accurately in diverse systems. These data are made available through various types of online resources. Connecting these online resources, public databases, services or simply static files,

[24] http://www.mutadatabase.org/
[25] http://globin.bx.psu.edu/phencode/
[26] http://www.gen2phen.org/

leverages a complexity increase in the implicit integration tasks. New problems revolve around integration and interoperability. Solving these problems is not trivial and, despite the fact that there are several on-going research projects in this area, computer science researchers have not yet discovered an optimal solution.

5.1.3 Locus-specific Databases

MUTbase[27] is the oldest of these tools and encompasses several Web databases for advanced genetic variation studies. The focus is toward mutation structural organization and availability to the community of both database curators and generic life scientists. Similarly, UMD[28] defines a structure and back-office for the management of variants. Although variants are usually publicly available, access to some genes is limited. Unlike these systems, LOVD[29] innovates with the "LSDB-in-a-box" approach. It is offered as a downloadable software package containing the full set of tools required for the deployment of a local locus-specific database.

This LSDB software diversity results in an extremely heterogeneous and fragmented network of independent data-rich silos, each with its own format and structure, with no interactions with other LSDBs or with any central systems. Consequently, this is a major drawback for data exchanges, aggregation in external systems or integration of resources. Available LSDB software frequently lacks semantic and contextual layers, resulting in datasets with no connections with external resources such as associated proteins or related metabolic pathways. Additionally, these systems were developed with specific gene curators in mind, meaning that the set of available features and included data are of specific relevance to curators, but neglects possible interested scientists, whether they are clinicians or biologists.

A need for distinct software tools, capable of tackling the combination of problems in both modern and legacy LSDBs, arises. A new approach must be adopted, accommodating not only gene and variant information, but also connections to external resources like proteins, diseases, publications or drugs. Furthermore, the aggregation of all available LSDBs (and their datasets) in a single central system should also be accomplished, providing a general vision over the entire genomic variation landscape.

[27] http://bioinf.uta.fi/MUTbase/

[28] http://www.umd.be/

[29] http://www.lovd.nl/

5.1.4 Gathering G2P Data

Considering the current status of the human variome research field, available systems, and what users expect from next- generation applications, four major challenges can be highlighted.

1. The aggregation of genetic variation datasets available in distributed and heterogeneous LSDBs is a critical step to improve the human variome research field.
2. Integration and interoperability of LSDBs should not be neglected and will play a key role in future systems. Therefore, any new approach must be prepared for further developments in the LSDB ecosystem.
3. Although current LSDBs are extremely rich data sources for curators, their limited scope undermines their adoption in the life sciences community, whose users expect additional extensive information alongside genes and their variants.
4. Accreditation, including authorship, ownership, and appropriate attribution are curators' major concerns. Hence, developments in this domain must take this into account, by displaying external content without devaluing the original systems from which the content has been aggregated.

These challenges require the need for an application like WAVe, based on an innovative strategy that is able to break with existing application design concepts and focus on extensibility, lightweight data integration, interoperability, and agile user interactions. From a computer science perspective, this represents a standard data integration problem. Available data are scattered through distributed and heterogeneous data sources, each with its own internal data collection strategies and severely lacking interoperability features or services.

5.1.5 Requirements Analysis and Design Issues

With high-demand for innovative human variome research software solutions and the challenges for gathering genotype-to-phenotype data, a set of general requirements and design strategies to determine WAVe's shape and structure were identified. This study resulted in six broad requirements, which include (R1) genetic dataset aggregation, (R2) genotype-to-phenotype integration, (R3) content accreditation, (R4) genetic data visual exploration, (R5) availability and (R6) exchange with software tools, each introduced next:

- **(R1) Genetic dataset aggregation.** The new WAVe platform must encompass new strategies for the aggregation of genetic variation datasets into a single centralized knowledge base.
    - **(R1.1) LSDB data extraction.** Considering the LSDBs landscape, WAVe must include methods for extracting variation data from MUTbase, UMD and LOVD, as well as from a variety of legacy systems.
- **(R2) Genotype-to-phenotype integration.** Innovative algorithms must be devised to integrate information regarding the genotype and the phenotype in a unique knowledge base.
    - **(R2.1) Data enrichment.** Aggregated genetic datasets must be enriched with direct pointers to a set of external applications relevant for the human variome field.
- **(R3) Content accreditation.** Original external applications must be displayed to users whenever possible. This will enforce the correct accreditation of content integrated within WAVe.
    - **(R3.1) Promote authorship, ownership and attribution.** Genetic variation data must always be linked to the original source, maintaining the gene curators' relevance within the research workflow.
- **(R4) Genetic data visual exploration.** Data for genes and variants must be presented in an easy-to-use interface, facilitating data navigation.
    - **(R4.1) Gene listing and browsing.** Access to genes must be direct through advanced searches or listings.
    - **(R4.2) Variation listing and browsing.** Access to aggregated variants must be provided in a unique summary, integrating all variants for a single gene in the same view.
- **(R5) Availability.** WAVe must be publicly available at all times for all users within and beyond the GEN2PHEN project context.
    - **(R5.1) Highly interactive web-based workspace.** WAVe's web interface must adopt modern user interaction strategies to deliver a highly interactive gene data exploration experience.
- **(R6) Exchange with software tools.** In addition (R5), acquired data should be made available to external software tools through a set of programming interfaces.

- **(R6.1) Interoperability API.** WAVe must include an interoperability API to enable access to rich gene and variant data for both software developers and skilled researchers.

This set of broad requirements drives the development of a new strategy to tackle the combination of miscellaneous challenges highlighted in past literature and summarized in Table 5-1.

Table 5-1. Mapping for the identified requirements with the challenges highlighted in relevant past publications on the human variome research field.

	R1	R2	R3	R4	R5	R6
Thorisson *et al.* [170]	●	●			●	●
Hawkins *et al.* [171]		●		●	●	
Muers *et al.* [172]	●	●				●
Mitropoulou *et al.* [173]	●		●	●		●
den Dunnen *et al.* [174]	●		●			
den Dunnen *et al.* [181]	●	●				●
Wu *et al.* [182]	●			●	●	●
Li *et al.* [183]		●			●	●

5.2 WAVe: Web Analysis of the Variome

5.2.1 Application Setup

Database Design

The WAVe platform is supported by a model centred on genes and their variants, which can be further extended with connections to miscellaneous resources, as shown in Figure 5-1. Starting with the general **Entity** concept, we specialize the **Gene** and **Variant** concepts. Taking in account the needs for external connections, one could also specify a concept for each integrated resource. However, this approach would lead to a static system, troubling future updates or the addition of novel relationships. With **Gene** and **Variant** concepts at the core of the model, and a dynamic relationship model, extensions can be added through the creation of new relationship types. As such, the model requires additional concepts: relationship type, entity and relationship. The **relationship type** indicates the type of additional information developers want to include in the application; the **entity** concept is used to store the values obtained from external resources, defining its main attributes along with the default **Gene** and **Variant** information; and the relationship concept is used to map elements from the core system to the new entities.

The main outcome of this approach is an extremely scalable model: it enables the configuration of any kind of relationship type, such as user mappings, external identifiers, gene properties or other required data type. Considering that the relationship values will be interpreted by the application, defining the entity values as strings maintains the system consistency, regardless of the inherent data type: string, integer, float or even boolean. Currently, this scheme is enough to support the inclusion of a broad range and large number of extensions including links to external applications or mappings to data types associated with genes, such as proteins, pathways or diseases. Moreover, new extensions can be added to the core concepts without breaking the platform workflow.

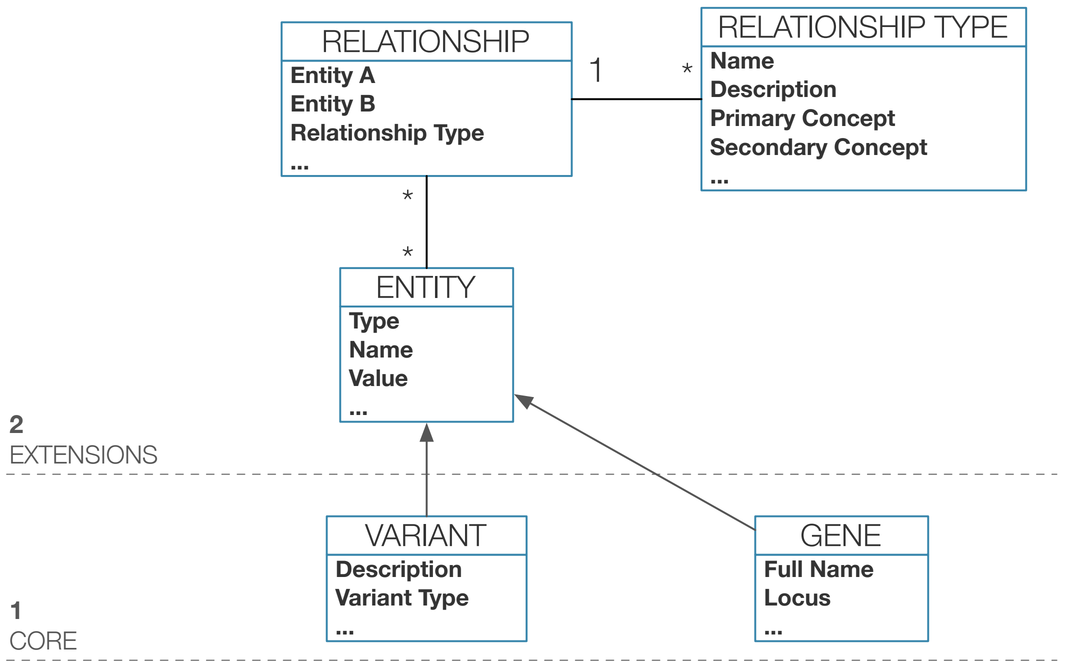


Figure 5-1. 1) Core (Gene and Variant) abstraction used for WAVe's database backend. 2) Extensions (Entity, Relationship and Relationship Type).

Application User Interface

Along with the various modelling considerations, WAVe also required designing the user interface. For this, miscellaneous mock-up interfaces were constructed focusing on several key interactions such as the gene navigation tree or the gene workspace. Independent users evaluated the initial mock-ups (Figure 5-2) and the resulting feedback was used to improve the final application interface, removing some real estate clutter and simplify the interface.

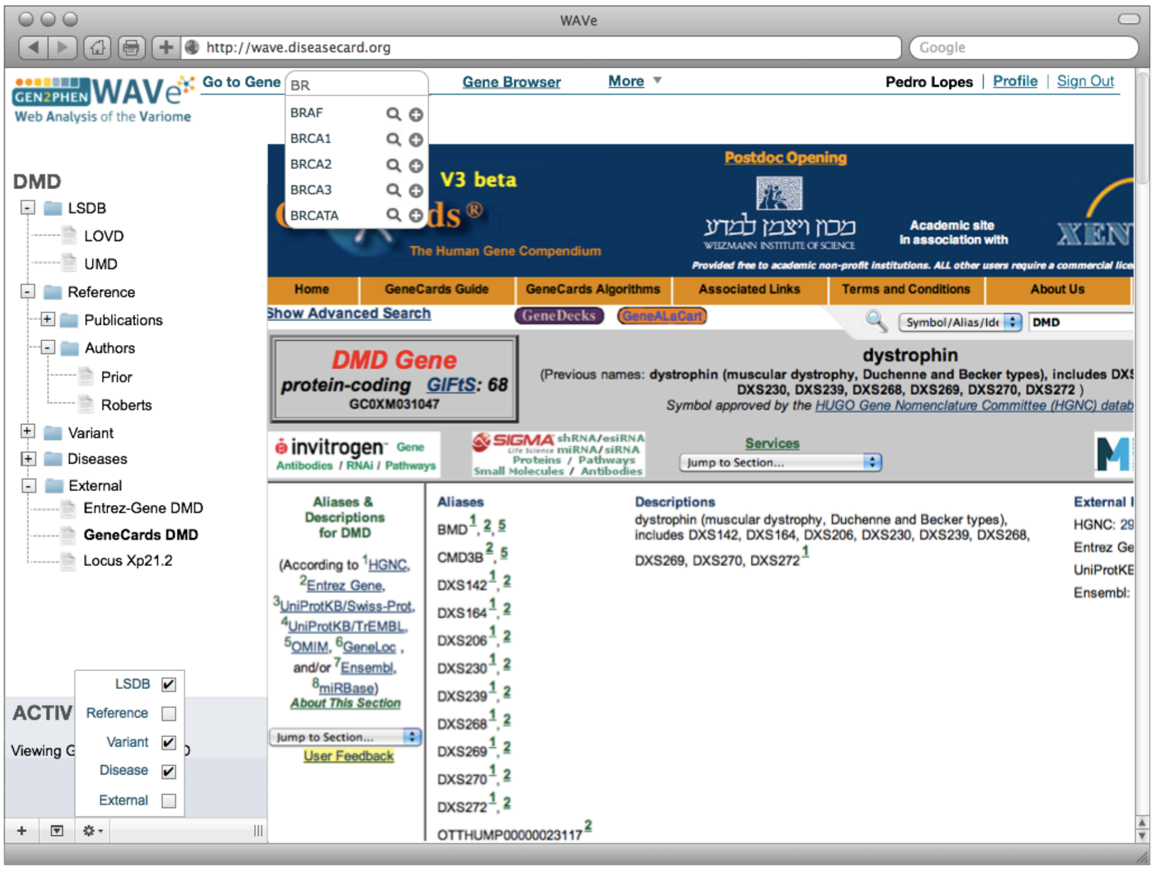


Figure 5-2. **Initial WAVe mock-up for the gene workspace, showing the GeneCards application for the human DMD gene.**

Architecture

WAVe's architecture is built around five components: a configuration file, the build engine, resource connectors, a database, and client applications. A diagram for component descriptions and interactions is shown in Figure 5-3.

The XML configuration file is composed of two parts: a static section, to store the relationship types and a modular section, to define the sources from where content for each relationship type will be extracted. This division is required to enable two database population moments: one to populate the **relationship types**, and another to add the **relationship** individuals, which can be replicated multiple times on distinct setups, enriching the original variation dataset. **Relationship type** configuration requires the name, description and identification of the primary and secondary connected concepts. Some sample configurations are shown further in this chapter.

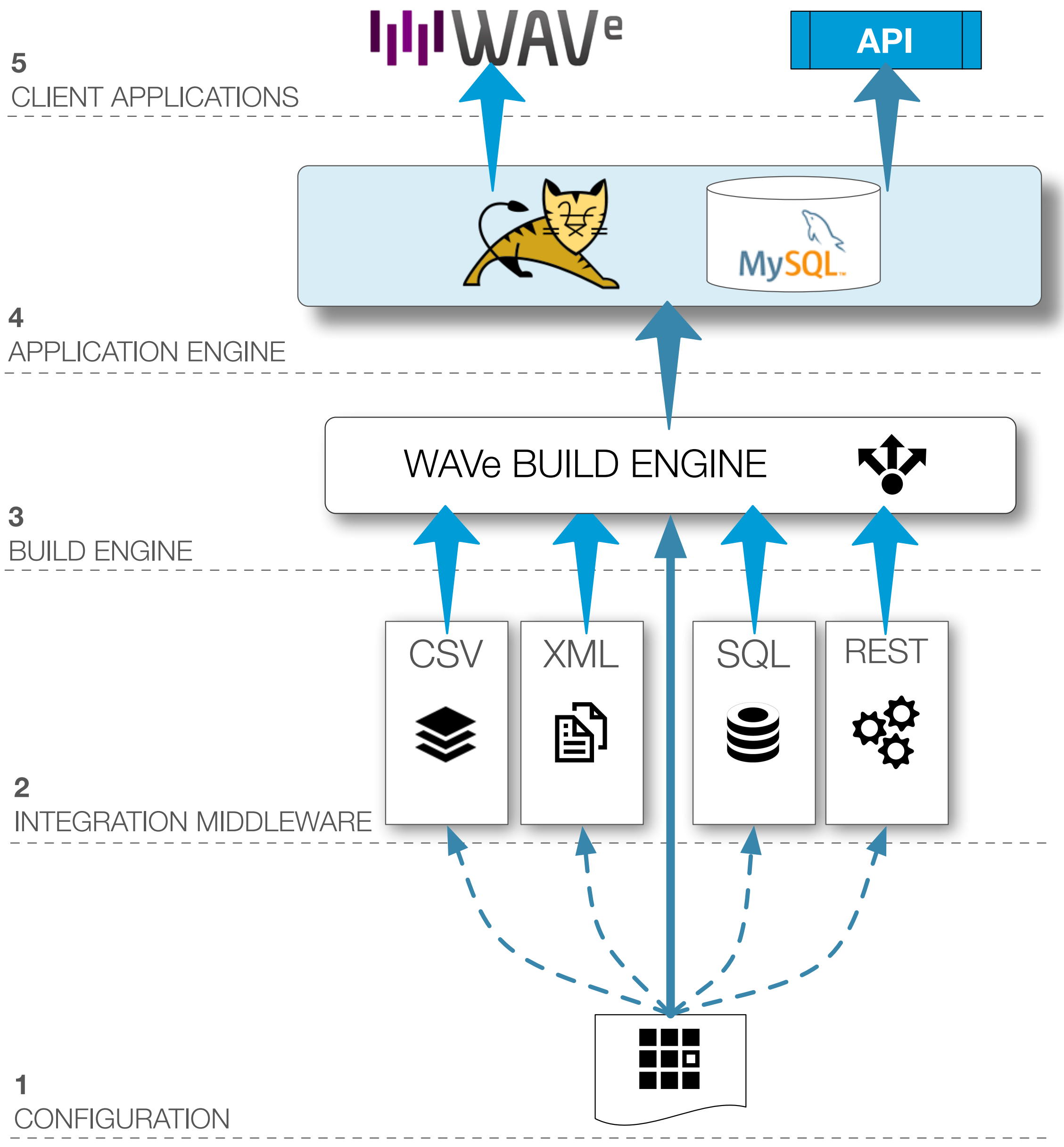


Figure 5-3. WAVe's architecture relies on five components. (1) The XML configuration file containing the system setup and the settings used to load genes, variants, and extensions data into WAVe's database. (2) Multiple resource connectors, used to wrap and read information for the build engine; these connectors enable loading data from CSV, XML, SQL, or REST Web services. (3) The build engine is responsible for processing the configuration file, reading WAVe's settings and loading data for the core and extensions according to the defined data sources. (4) WAVe's application engine includes the server-side code and the database, which is populated by the build engine and accessed by client applications. The database replicates Gene and Variant information while extensions are loaded as simple pointers to external applications. (5) Client applications are WAVe's entry point.

Each **relationship type** can have several associated data sources and each data source may have its own data gathering methods. Therefore, and to increase even further the platform extensibility, the configuration file can accommodate four distinct methods to extract data from the sources. Each of these methods is associated with a specific type of extraction, thus requiring different configuration features. The CSV method allows the extraction of data from CSV files, and therefore requires definition of the file location, the column that should be read and the line where the reading process will start. The XML method requires a similar configuration: file location and an XPath expression to query the content from the file. Next, the SQL data extraction method requires a database connection string, a SQL query to select the content and the name of the column where the desired content will be included. At last, the REST connector requires the service address and an additional parameter defining the service reply format, XML or CSV. To parse web service replies, the build engine combines the required method, CSV or XML, with the web service method. Relying on these four data gathering methods improves the platform extensibility. In addition to the data gathering methods, configuring a data source requires a name and description plus the concept that will be passed as a variable to the build engine. The latter is required because the extraction is made based on the primary relationship concept. The configuration flexibility allows the system to be adapted to multiple contexts, applications and usage scenarios.

Finally, the build engine, implemented in Java, is responsible for reading the configuration file and loading the data to populate the platform database. The build engine comprises a set of tools for processing each of the possible loading methods. These integrative wrappers enable independent data replication from SQL, XML, CSV or REST services, and are WAVe's bridge from the external data sources to the internal database. The latter is a MySQL database, which was designed to include miscellaneous dataset versions, allowing for streamlined application and data updates.

For the application engine, the Java Stripes web framework was chosen. This library enhances the deployment of web applications, by adding an abstraction layer to Java's default web system. For example, with it, it is much easier to create custom application URLs and perform the binding to the desired Java bean. The application engine serves WAVe's web interface and API, which are detailed further in this chapter.

LSDB Data Extraction

Reading data from the myriad of existing LSDBs triggered various issues. Despite the majority of available LSDBs being built on top of the LOVD platform, other systems are still widely used. While access to UMD and Mutbase data could be streamlined, a couple hundred LSDBs remained out of the reach and required a custom variant import tool. The variant extraction workflow is overviewed in Figure 5-4.

At first, an empirical study was conducted to assess what variant data should in fact be extracted from each locus-specific database. After this, it was clear that WAVe needs to read HGVS-compliant variant descriptions. Since WAVe is gene-centric, when variants are being imported we know exactly to which gene they belong. Furthermore, from a complete variant description we can infer the analysed reference sequence, the type of variant and what/where the change has occurred. The following examples were obtained from COL3A1 LSDB and demonstrate this characteristic:

- NM_000090.3:c.413delC matches a single base-pair deletion, cytosine, at position 413 in the coding region defined at reference sequence NM_000090.3;
- NM_000090.3:c.2708G>A signals a single base-pair substitution, from guanine to adenine, at position 2708 in the coding region defined at reference sequence NM_000090.3.

As one can easily extrapolate, with the gene, the LSDB location and the variant description, there is enough information to organize a unique variation dataset and enable the browsing of collected data from multiple perspectives.

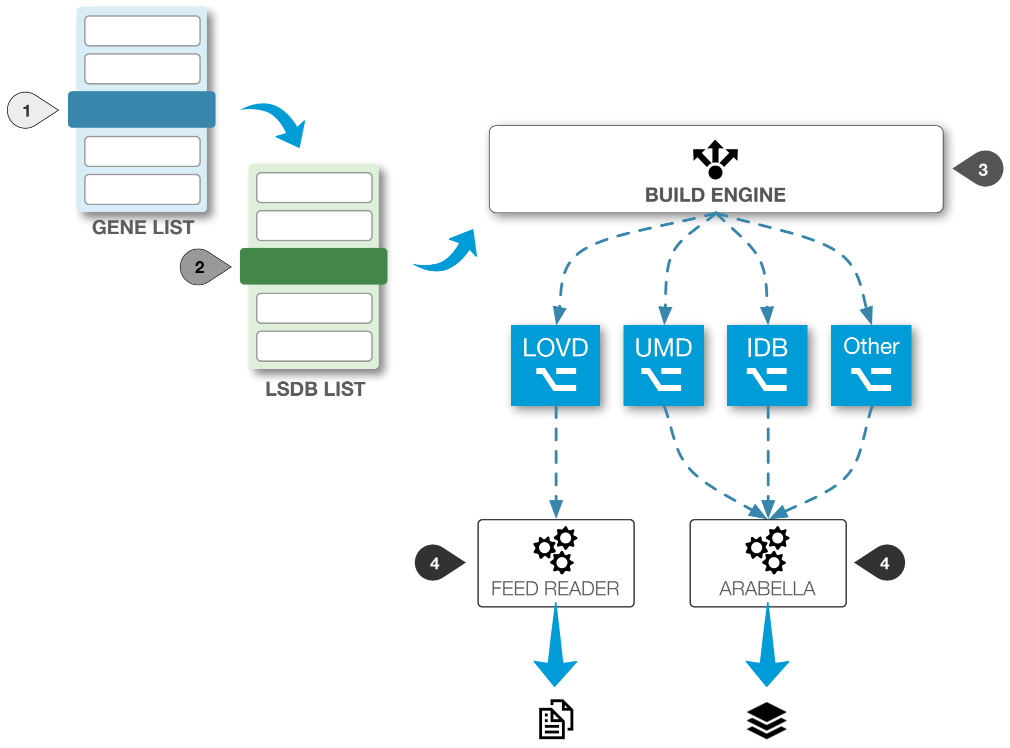


Figure 5-4. Variant import workflow. 1) Each gene from WAVe's gene list is processed iteratively. 2) The LSDB list for each gene is obtained and processed iteratively in the build engine. 3) The build engine launches the appropriate data import method for each LSDB type. 4) Data are read using a middleware feed reader for LOVD instances and using Arabella web crawler for the remaining LSDBs.

By having a well-defined description schema, variants can be easily understood and, more importantly, read using a set of regular expressions. Since there is no streamlined process of reading variants from legacy LSDBs, our variant import strategy revolves around using a targeted web crawler, directed at finding variants in HTML pages. Arabella [187], an in-house web crawler, was modified to identify variants in web pages and included in WAVe. This tool is injected with a list of LSDB URLs for each gene and retrieves a XML container with the found variants. With UMD and IDbases lacking web services, Arabella is also used to read variants from the public LSDBs built on top of these platforms.

Extracting data from LOVD systems revealed to be an easier process. Starting with version 2.0, LOVD includes a public API composed of a public web service that lists variants from each LSDB instance in Atom feed format[30]. This LOVD interoperability feature meant that to integrate all variants from LOVD systems a simple feed reader and respective parser was required.

Enriching G2P Data

WAVe's data integration pipeline completes with the data enrichment process. Once genes, LSDBs and variants are loaded into WAVe's knowledge base, the build engine starts loading data from external resources, according to their descriptions in the configuration file. As mentioned in the application setup, this data integration process is done using a WAVe-specific middleware wrapper. As WAVe is a gene-centric platform, the build engine uses the gene HGNC symbol to identify external identifiers and load them into WAVe's knowledge base. Next, we describe the usage of these integration wrappers within WAVe.

The first scenario involves loading relationships to UniProt identifiers. UniProt data are available in various formats; hence, CSV was chosen to select the UniProt identifiers associated with the gene WAVe is reading. The configuration for this integrative wrapper is as follows:

```
    <!—UniProt CSV wrapper configuration -->
    <value>
        <name>UniProt/SwissProt</name>
        <description>Information regarding available, active and reviewed proteins
in UniProt.</description>
        <shortname>SwissProt</shortname>
        <source>
                <method>cache</method>
                <type>csv</type>
                <connection>http://www.uniprot.org/uniprot/?query=gene_exact:#replac
        eme#+AND+active:yes+AND+reviewed:yes+AND+organism:9606&amp;format=tab&amp;c
        olumns=id</connection>
```

[30] Sample feed URL for COL3A1 gene variants: https://eds.gene.le.ac.uk/api/rest.php/variants/COL3A1

```
            <query>\t</query>
            <result>0</result>
      </source>
      <value>http://www.uniprot.org/uniprot/#replaceme#</value>
      <parent></parent>
      <type>protein</type>
      <ua>uniprot</ua>
</value>
```

The configuration includes all fields required by both WAVe's build engine and application engine. The **name**, **description** and **shortname** properties are used in WAVe's web interface to fill in each external concept metadata. Next, the **source** property set includes information for WAVe's build engine: **method** defines the data loading method being used; **type** defines the external data type and is used to select the proper wrapper during the build process; **connection** states the data connection string or, in this case, the CSV file location; **query** defines the CSV delimiter and **result** configures the CSV starting line. Note that the *#replaceme#* string component in the **connection** property is replaced with the gene HGNC symbol. This last substitution is what filters UniProt web service results, listing only the identifiers associated with one particular gene. The application engine uses the **value** property to compose the external resource URL. This process creates a valid URL combining the identifiers loaded in the data import process with the **value** property content. The **type** property defines to which entity, in the web interface, these data will belong to. At last, the **ua** value defines the UniversalAccess identifier keyword to be used in WAVe's API, detailed further in this document.

GeNS is an in-house data warehouse that contains more than 100 million identifiers and mappings for life sciences databases [79]. With such a huge in-house resource, we decided to use it as a supplier for most of external resources relationships. Next is a sample GeNS resource configuration to load KEGG database mappings to a specific gene HGNC symbol.

```
<!-- KEGG SQL wrapper configuration -->
<value>
      <name>KEGG</name>
      <description>Information from metabolic pathways available in Kyoto
Encyclopedia of Genes and Genomes database.</description>
      <shortname>KEGG</shortname>
       <source>
            <method>cache</method>
            <type>sql</type>
            <connection>jdbc:sqlserver://sql2k8-
      ua.servers.ua.pt;database=GeNS;user=*****;password=*****</connection>
            <query>SELECT DISTINCT I.Alias AS result FROM Identifier  I WHERE
      I.DataTypeId = 11 AND I.ProteinId IN (SELECT P.ProteinId FROM Protein P
      INNER JOIN Identifier I ON I.ProteinId = P.ProteinId WHERE P.TaxonomicId =
      9606 AND I.DataTypeId = 1 AND I.Alias LIKE '#replaceme#')</query>
            <result>result</result>
```

```
        </source>
        <value>http://www.genome.jp/dbget-bin/www_bget?#replaceme#</value>
        <type>pathway</type>
        <ua>kegg</ua>
    </value>
```

As one can easily assess, most configuration properties are very similar between CSV and SQL data resources. The **type** property is, obviously, set to *sql* and the **connection** property contains an actual Java JDBC connection string. The main difference to the UniProt CSV scenario distinctions lie in the **query** property, which now contains a full SQL query, and in the new **result** property, stating the column name from where results will be read. Whereas in the CSV definition, the gene symbol replacement took place in the **connection** property, in the SQL wrapper this substitution occurs in the **query** property.

In some cases, a particular connection to an external resource is obtained directly using the gene HGNC symbol. The next configuration sample details the integration of GeneCard's resource.

```
    <!-- GeneCards direct method configuration -->
    <value>
        <name>Gene Cards</name>
        <description>Information from Gene Cards.</description>
        <shortname>GeneCards</shortname>
        <source>
               <method>direct</method>
        </source>
        <value>http://www.genecards.org/cgi-
        bin/carddisp.pl?gene=#replaceme#</value>
        <type>locus</type>
        <ua>genecard</ua>
    </value>
```

In these configuration properties, we have to highlight the lighter **source** property set. In these cases, just defining the **method** property as *direct* is enough to inform the build engine that no information should be imported. Therefore, only the application engine will use these configuration properties to full effect.

The strategy of having abstract integration middleware wrappers enables a more general and streamlined data integration process. With a few configuration settings, one can easily establish connections to external resources and replicate specific data to a core data warehouse. This is a leap forward from the proposal introduced with DynamicFlow, where the description were simpler and, consequently, less powerful.

5.2.2 Data Content and Usefulness

WAVe delivers integrated access to miscellaneous online resources. To cope with the immense resource diversity, WAVe's lightweight integration mechanism plays a key role.

In a simple hypothetical scenario, a need to establish relations between genes and clinical trials is identified. The United States National Institutes of Health Clinical Trials[31] database is defined as the main data source, and identifiers are quickly loaded into WAVe's database, creating new relations between a gene and its available clinical trials. WAVe will store the unique clinical trial identifiers and a dynamic URL to access each clinical trial web page within WAVe. This approach allows WAVe to store both clinical trials data and pointers to gathered Clinical Trials web pages. Consequently, one can identify clinical trials associated with each gene and access each Clinical Trial in WAVe's gene workspace.

Core Genetics Datasets

The LSDB list used in WAVe is maintained in cooperation with GEN2PHEN project partners[32]. This list can be integrated in any application or downloaded for personal use. Figure 5-5 shows the distribution of LSDBs according to their type. Clearly, LOVD is the most widely used LSDB and the greater contributor to WAVe's dataset.

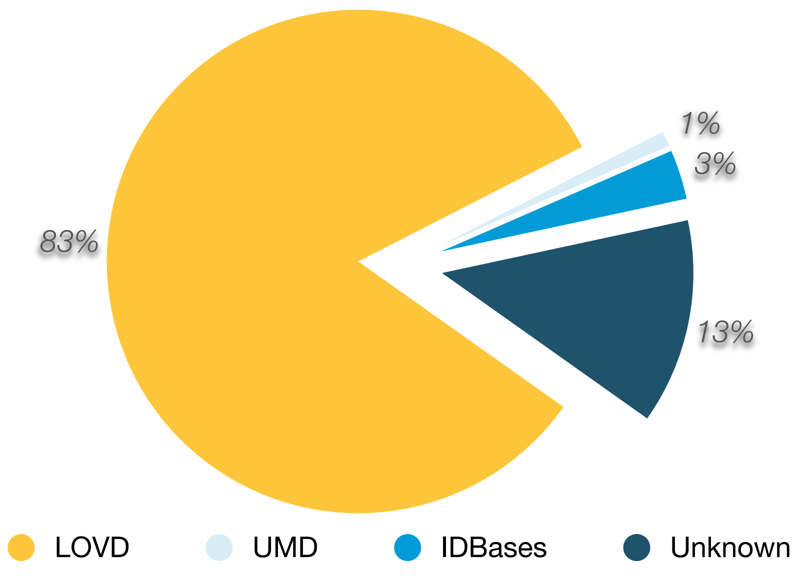


Figure 5-5. Locus-specific database distribution in WAVe and GEN2PHEN's list. LOVD 83%, Unknown 13%, IDBases 3%, UMD 1%.

The HGNC gene list[33] can be downloaded and used in various ways. This list is maintained by the US National Human Genome Research Institute (NHGRI) and the Wellcome Trust and represents the most up to date list of valid and known gene symbols and names, along with other miscellaneous identifiers.

[31] http://www.clinicaltrials.gov/
[32] http://www.gen2phen.org/data/lsdbs/
[33] http://www.genenames.org/

Table 5-2 lists WAVe's extensions and respective data sources. WAVe currently features 10 data types, linking 20 distinct resources through more than 500,000 pointers and identifiers.

Table 5-2. WAVe's extensions and corresponding external resources with their respective base URLs.

EXTENSION	RESOURCE	ORIGINAL URL
LSDB	UMD	http://www.umd.be/
	IDBases	http://bioinf.uta.fi/
	LOVD	http://www.lovd.nl/
Gene	GeneCards	http://www.genecards.org/
	HGNC	http://www.hgnc.org/
	Entrez	http://www.ncbi.nlm.nih.gov/gene/
Publication	QuExT	http://bioinformatics.ua.pt/quext/
	Pubmed	http://www.ncbi.nlm.nih.gov/pubmed/
Disease	OMIM	http://www.ncbi.nlm.nih.gov/omim/
Pharmacogenomics	PharmGKB	http://www.pharmgkb.org/
Locus	MapViewer	http://www.ncbi.nlm.nih.gov/projects/mapview/
	Ensembl	http://www.ensembl.org/
Pathway	KEGG	http://www.genome.jp/kegg/
	REACTOME	http://www.reactome.org/
Protein	UniProt/SwissProt	http://www.uniprot.org/
	UniProt/TrEMBL	http://www.uniprot.org/
	PDB	http://www.pdb.org/
	Expasy	http://expasy.org/
	InterPro	http://www.ebi.ac.uk/interpro/
Ontology	GO	http://amigo.geneontology.org/

5.2.3 Case Study

To assess WAVe's applicability in a research workflow, a data-mining scenario was studied. A biologist searching for information regarding the COL3A1 (collagen, type III, alpha 1) gene might need to answer the following questions:

1. Are there any LSDBs for the human COL3A1 gene? Where can they be accessed, and who are the curators?
2. If such LSDBs exist, what are the known variants for COL3A1 and where were they published?

3. What are the diseases associated with COL3A1 and what are the most relevant publications regarding these diseases?
4. Has anybody developed drugs for diseases associated with COL3A1 variants?
5. What protein information is there for COL3A1?
6. What are the metabolic pathways related to COL3A1?
7. Does any genome browser provide information about COL3A1?
8. Are there specific Gene Ontology terms that relate to COL3A1?

Without an integrated query environment such as WAVe, the solution relies on the biologist accessing multiple applications, in an ad hoc fashion, until all the questions are answered. The starting point might be the GEN2PHEN-maintained list of locus specific mutation databases: searching for "COL3A1" yields two databases links, either of which must be followed to access the recorded variants. Next, to find COL3A1-related protein information, the biologist needs to access UniProt, query the database for "COL3A1" human information, filter the results and then access the 'reviewed proteins. For pathways, the required steps are similar: access the KEGG website, query for "COL3A1" and filter for the human species. The iterative process continues until a large number of applications have been accessed, resulting in multiple open browser windows or tabs, each involving some manner of querying and filtering until the data are finally accessed. The total number of individual user actions is large.

Using WAVe, a biologist simply searches for the "COL3A1" gene and the workspace immediately displays all of the required information[34]. The answers to the initial set of questions are:

1. There are two LSDBs for COL3A1. A publicly accessible LOVD instance (http://bioinformatics.ua.pt/WAVe/gene/COL3A1/lsdb:726) and a UMD instance (http://bioinformatics.ua.pt/WAVe/gene/COL3A1/lsdb:725), which is protected by a username and password.
2. Variants for COL3A1 can be found under the '**Variation'** node of WAVe's navigation tree (http://bioinformatics.ua.pt/WAVe/gene/COL3A1/variantall:COL3A1) or in the **Variants** tab of the available LOVD LSDB.
3. WAVe's **Disease** node shows three NCBI OMIM entries associated with COL3A1: MIM 120180 (Collagen, Type III, Alpha-1; COL3A1), MIM 130020 (Ehlers-Danlos

[34] http://bioinformatics.ua.pt/WAVe/gene/COL3A1/

Syndrome, Type III) and MIM 130050 (Ehlers-Danlos Syndrome, Type IV, Autosomal Dominant). The most relevant publications are accessed through QuExT [188] (http://bioinformatics.ua.pt/WAVe/gene/COL3A1/quext:COL3A1) and PubMed (http://bioinformatics.ua.pt/WAVe/gene/COL3A1/pubmed:COL3A1), under the **Publication** node.

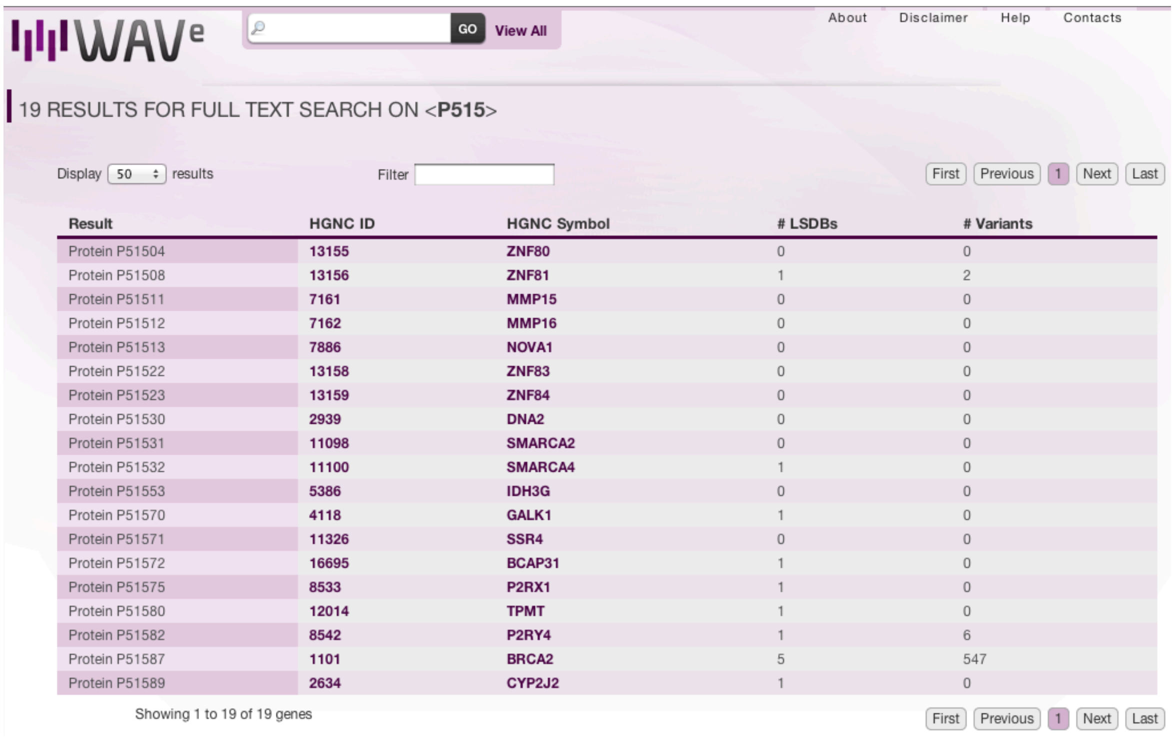
4. The **Pharmacogenomics** node shows that PharmGKB has one entry associated with the COL3A1 gene, where *letrozole* and *orbofiban* are mentioned (http://bioinformatics.ua.pt/WAVe/gene/COL3A1/pharmgkb:PA26716).
5. The **Protein** node provides access to UniProt/SwissProt leading directly to the COL3A1 entry (http://bioinformatics.ua.pt/WAVe/gene/COL3A1/uniprot:P02461).
6. WAVe integrates metabolic pathways from both KEGG and Reactome. For COL3A1, only KEGG has information for four pathways: *focal adhesion*, *ECM-receptor interaction*, *protein digestion and absorption*, and *amoebiasis* (http://bioinformatics.ua.pt/WAVe/gene/COL3A1/kegg:hsa:1281).
7. COL3A1 is available in the **Genome** node in NCBI's MapViewer (http://bioinformatics.ua.pt/WAVe/gene/COL3A1/mapview:COL3A1) and Ensembl (http://bioinformatics.ua.pt/WAVe/gene/COL3A1/ensembl:ENSG00000168542).
8. The results displayed in the **Ontology** node indicate that COL3A1 is involved in two distinct molecular functions, 12 biological processes, and two cellular components.

WAVe provides much more information than might be achieved by *ad hoc* searching, but does so with much less user interaction and in less time. In summary, using standalone applications is an inefficient approach to address the initial data-mining problem, although WAVe proves its efficacy by offering integrated access to curated and distributed data, irrespective of type or location.

5.2.4 Features and Usability

Search and Browse

Searching on the homepage or in the top search box triggers the automatic suggestion engine, displaying results matching users' input. In addition to gene searches, based on HGNC gene symbols, genes can be obtained through their association with a set of common and well-known identifiers. Users can search for UniProt, OMIM, GWASCentral, KEGG, Reactome, PharmGKB, or Gene Ontology identifiers, and WAVe displays the list of genes related with the queried entity. Search results are displayed in the **browse** interface (Figure 5-6).



19 RESULTS FOR FULL TEXT SEARCH ON <**P515**>

Result	HGNC ID	HGNC Symbol	# LSDBs	# Variants
Protein P51504	13155	ZNF80	0	0
Protein P51508	13156	ZNF81	1	2
Protein P51511	7161	MMP15	0	0
Protein P51512	7162	MMP16	0	0
Protein P51513	7886	NOVA1	0	0
Protein P51522	13158	ZNF83	0	0
Protein P51523	13159	ZNF84	0	0
Protein P51530	2939	DNA2	0	0
Protein P51531	11098	SMARCA2	0	0
Protein P51532	11100	SMARCA4	1	0
Protein P51553	5386	IDH3G	0	0
Protein P51570	4118	GALK1	1	0
Protein P51571	11326	SSR4	0	0
Protein P51572	16695	BCAP31	1	0
Protein P51575	8533	P2RX1	1	0
Protein P51580	12014	TPMT	1	0
Protein P51582	8542	P2RY4	1	6
Protein P51587	1101	BRCA2	5	547
Protein P51589	2634	CYP2J2	1	0

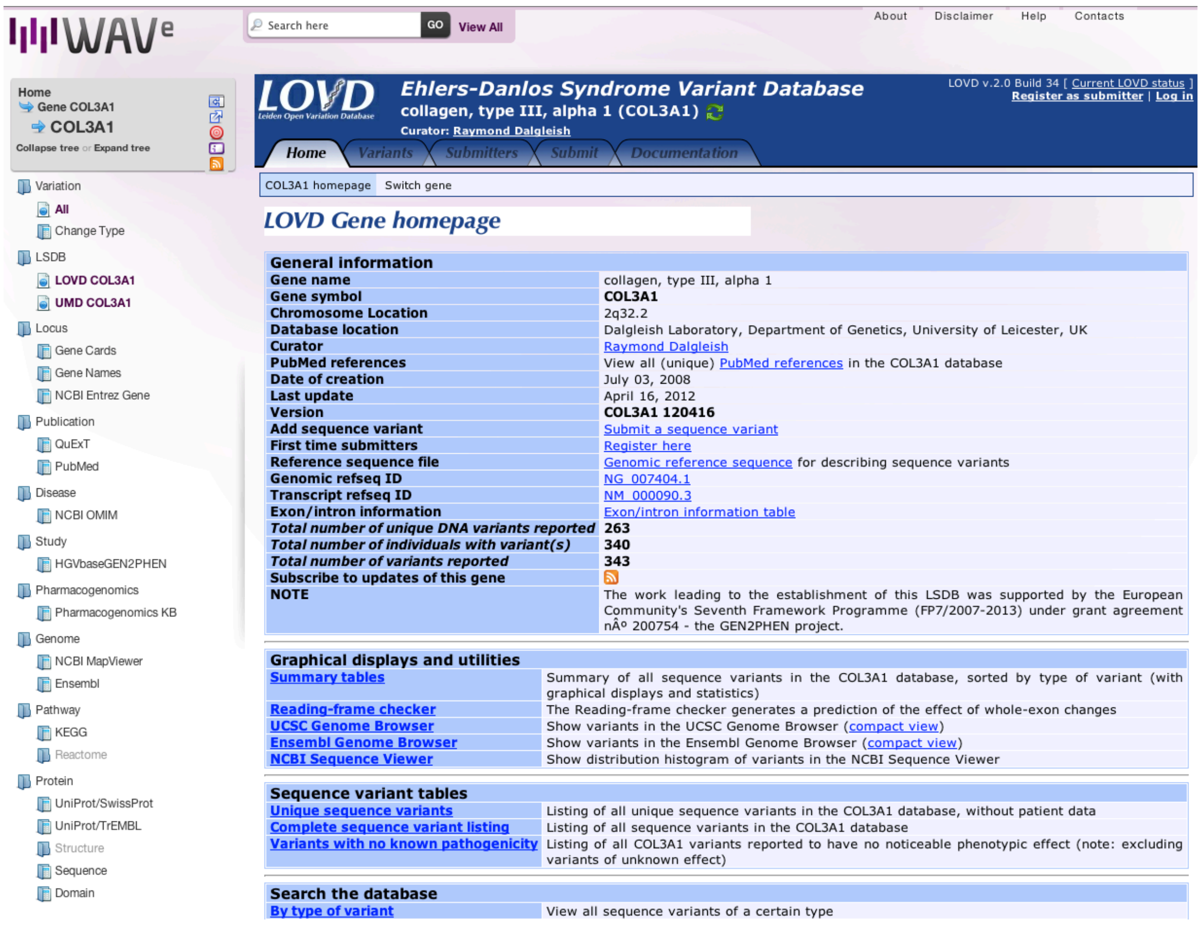
Showing 1 to 19 of 19 genes

Figure 5-6. WAVe browse interface with search results for the P515 query. The result set originates a Gene Mesh that is used to keep users in context of their queries in the gene workspace.

This interface lists all genes having known associations with the queried term. This associated genes list, the **Gene Mesh**, is further available in the gene workspace toolbox. Browsing results can be further filtered through the **Filter** box. This performs another search within the page scope, that is, within the **Gene Mesh**. At last, searching for "*" lists all available genes and their respective LSDB and variant count. At this interface, users can remove genes without LSDB through the top toggle button.

Workspace

The gene workspace is the main data exploration area. This is where users can access collected LSDBs and respective variants or navigate through WAVe's rich relationship dataset. WAVe provides a holistic view over the human variome research domain, connecting a multitude of distinct data types and resources, in a coherent interface, without limiting application usability (Figure 5-7).



Ehlers-Danlos Syndrome Variant Database
collagen, type III, alpha 1 (COL3A1)
Curator: Raymond Dalgleish

LOVD Gene homepage

General information	
Gene name	collagen, type III, alpha 1
Gene symbol	COL3A1
Chromosome Location	2q32.2
Database location	Dalgleish Laboratory, Department of Genetics, University of Leicester, UK
Curator	Raymond Dalgleish
PubMed references	View all (unique) PubMed references in the COL3A1 database
Date of creation	July 03, 2008
Last update	April 16, 2012
Version	COL3A1 120416
Add sequence variant	Submit a sequence variant
First time submitters	Register here
Reference sequence file	Genomic reference sequence for describing sequence variants
Genomic refseq ID	NG_007404.1
Transcript refseq ID	NM_000090.3
Exon/intron information	Exon/intron information table
Total number of unique DNA variants reported	263
Total number of individuals with variant(s)	340
Total number of variants reported	343
Subscribe to updates of this gene	
NOTE	The work leading to the establishment of this LSDB was supported by the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement nÂº 200754 - the GEN2PHEN project.

Graphical displays and utilities	
Summary tables	Summary of all sequence variants in the COL3A1 database, sorted by type of variant (with graphical displays and statistics)
Reading-frame checker	The Reading-frame checker generates a prediction of the effect of whole-exon changes
UCSC Genome Browser	Show variants in the UCSC Genome Browser (compact view)
Ensembl Genome Browser	Show variants in the Ensembl Genome Browser (compact view)
NCBI Sequence Viewer	Show distribution histogram of variants in the NCBI Sequence Viewer

Sequence variant tables	
Unique sequence variants	Listing of all unique sequence variants in the COL3A1 database, without patient data
Complete sequence variant listing	Listing of all sequence variants in the COL3A1 database
Variants with no known pathogenicity	Listing of all COL3A1 variants reported to have no noticeable phenotypic effect (note: excluding variants of unknown effect)

Search the database	
By type of variant	View all sequence variants of a certain type

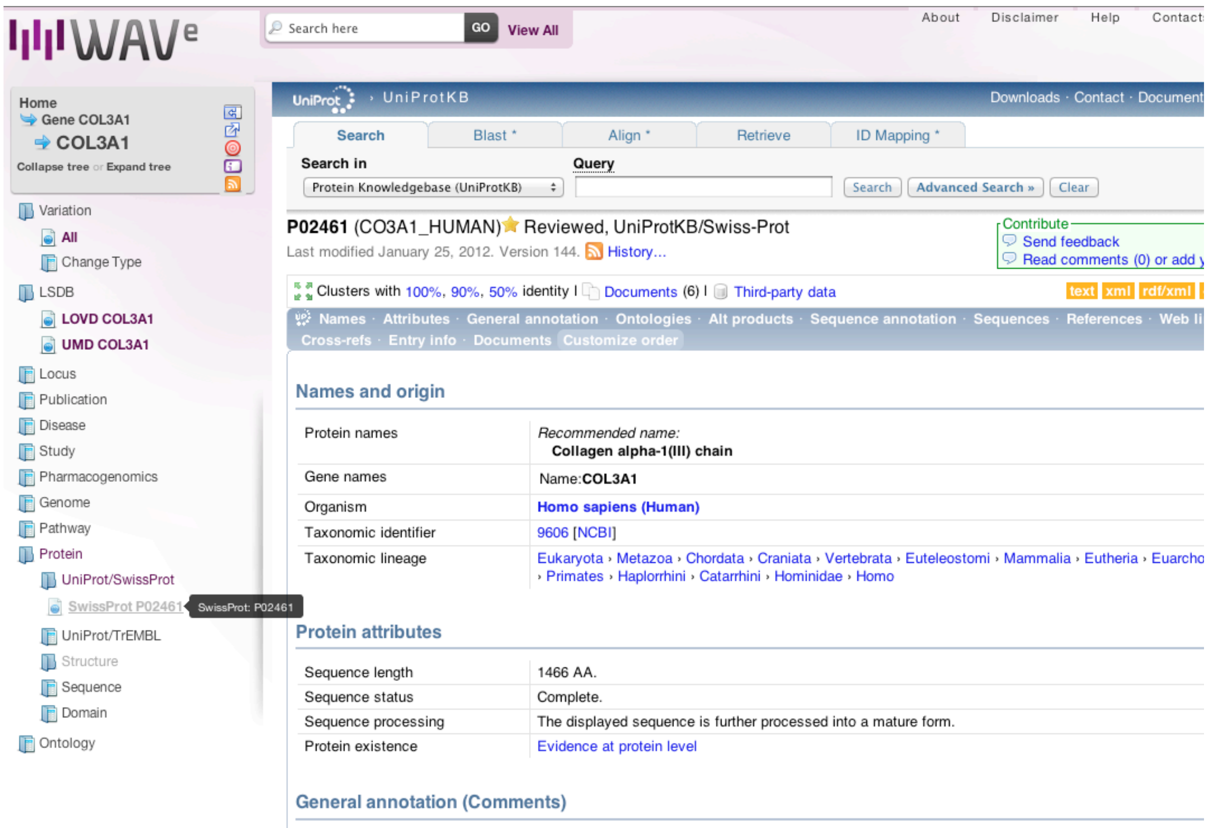
Figure 5-7. WAVe gene analysis workspace interface for the human COL3A1 (collagen, type III, alpha 1) gene. Sidebar with the COL3A1 gene navigation tree: direct access to relevant gene-related information. Central area with WAVe's Live View **mode: external applications (in this case, LOVD installation for COL3A1) are loaded within WAVe's interface.**

WAVe is based on a directory navigation tree metaphor. The rationale behind this approach is that one can organize all available information in a dynamic tree, where each type of external resource corresponds to a principal node in the tree. Consequently, each child node in the tree will be linked to resources made available for a specific resource type, and each tree leaf will be the direct pointer to a resource instance. This results in a dynamic and extensible resource directory. For example, the **Locus** node includes access to three resources: GeneCards, HGNC's Gene Names and NCBI's Entrez Gene, each containing a leaf pointing to the connected resource: a Web application with gene information for the gene being analysed.

Along with the gene navigation tree, the left sidebar also includes a toolbox displaying action items for each gene. Their actions are, from top to bottom: expand the **Live View** mode to full screen, open the external resource in a new window, view other genes in the **Gene Mesh**, view WAVe's gene summary, and open the gene feed.

Live View

Each click on a leaf in WAVe's gene tree triggers the **Live View** mode, loading external resources in the workspace's central area (Figure 5-8).



P02461 (CO3A1_HUMAN) Reviewed, UniProtKB/Swiss-Prot
Last modified January 25, 2012. Version 144. History...

Clusters with 100%, 90%, 50% identity | Documents (6) | Third-party data

Names and origin

Protein names	*Recommended name:* **Collagen alpha-1(III) chain**
Gene names	Name:**COL3A1**
Organism	**Homo sapiens (Human)**
Taxonomic identifier	9606 [NCBI]
Taxonomic lineage	Eukaryota › Metazoa › Chordata › Craniata › Vertebrata › Euteleostomi › Mammalia › Eutheria › Euarcho › Primates › Haplorrhini › Catarrhini › Hominidae › Homo

Protein attributes

Sequence length	1466 AA.
Sequence status	Complete.
Sequence processing	The displayed sequence is further processed into a mature form.
Protein existence	Evidence at protein level

General annotation (Comments)

Figure 5-8. WAVe gene analysis workspace interface for the human COL3A1 (collagen, type III, alpha 1) gene, highlighting the UniProt entry P02461[35].

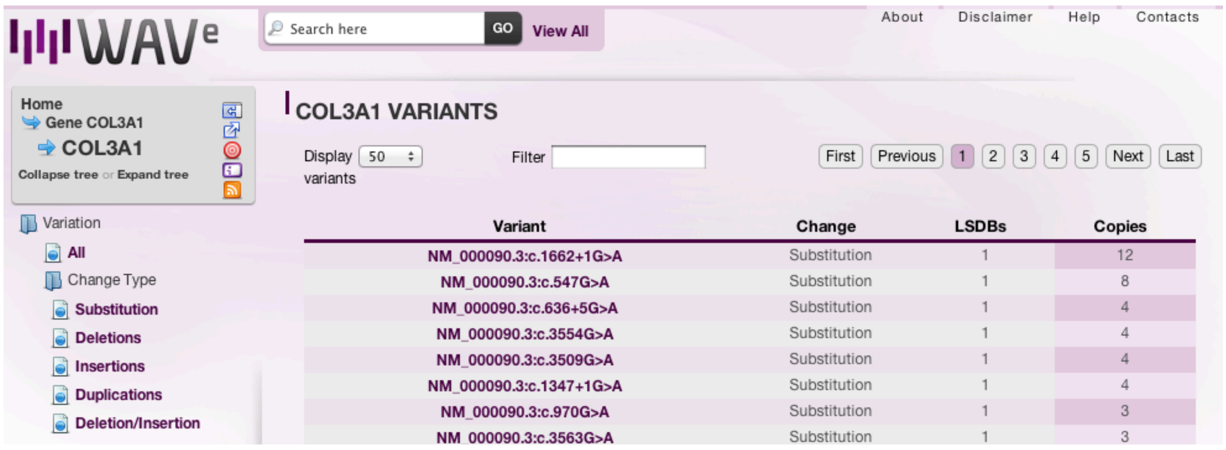
This feature enables loading external applications, such as UniProt, Entrez Gene, or any LSDB, inside WAVe's interface. A primary consequence is that WAVe users never lose the context of their ongoing search and are able to browse multiple distributed and heterogeneous resources without leaving WAVe. Furthermore, content authorship and ownership is also assured. Where most widely used applications collect content for custom

[35] http://www.uniprot.org/uniprot/P02461

interface display, WAVe directs the user to the original application, without compromising any of its features.

Variant Browser

WAVe's variant browser is a unique tool that provides direct access to distributed variation datasets in a single list view. These datasets are collected by WAVe from a multitude of LSDBs. Through the gene navigation tree (Figure 5-9), users can browse variations by change type (*Substitution*, *Deletion*, *Inversion*, *Insertion*, *Duplication* or *Deletion/Insertion*) or list all variants. Only change types with matching variants are displayed. For example, if no deletions are collected for a given gene, the deletion link is not displayed.



COL3A1 VARIANTS

Display 50 variants Filter

First Previous 1 2 3 4 5 Next Last

Variant	Change	LSDBs	Copies
NM_000090.3:c.1662+1G>A	Substitution	1	12
NM_000090.3:c.547G>A	Substitution	1	8
NM_000090.3:c.636+5G>A	Substitution	1	4
NM_000090.3:c.3554G>A	Substitution	1	4
NM_000090.3:c.3509G>A	Substitution	1	4
NM_000090.3:c.1347+1G>A	Substitution	1	4
NM_000090.3:c.970G>A	Substitution	1	3
NM_000090.3:c.3563G>A	Substitution	1	3

Figure 5-9. WAVe's variant listing interface. The Variation node in the sidebar provides quick access to the collected variant lists, sorted by available mutation change types. Variants are listed in a dynamic table in the central content area.

By clicking these leafs, the **Live View** features loads the variant browsing interface. This view displays a table including: the variant description in HGVS-compliant format; the used reference sequence, where available; the variant type, such as **Substitution** or **Deletion**; the number of LSDBs where the variant is listed; and the total number of variant copies. Clicking the variant description link leads to the original LSDB page or, in cases where there is more than one source, to the list of LSDBs containing the variant. Variants can also be searched through the **Filter** box. This allows searching for particular variants, locations of variant types.

WAVe API

Whilst WAVe's web user interface is its central access point, it also has an API. Interoperability was always a main concern during WAVe development and making integrated data available for further usage was a top priority. Therefore, WAVe's API provides aggregated data in Atom or JSON data-exchange formats. There was emphasis on providing content feeds, which are easily usable in any software development framework or readable in any feed reader. Currently, WAVe's API allows access to integrated resources listed in the gene navigation tree and to gene variants listing. With the latter, WAVe was the first platform to provide programmable access to variants aggregated from multiple, distributed, and heterogeneous locus-specific databases. Next, some of these methods are detailed.

→ Accessing Gene Data: *http://bioinformatics.ua.pt/WAVe/gene/<gene>/<format>*

- This method outputs the list of rich dataset links collected by WAVe. The **gene** keyword should be replaced by a valid HGNC symbol and **format** may be replaced by *atom* or *json*, for Atom feeds or JSON objects respectively.
- http://bioinformatics.ua.pt/WAVe/gene/COL3A1/json: COL3A1 data in JSON format.

→ Accessing Variant Data: *http://bioinformatics.ua.pt/WAVe/variant/<gene>/atom*

- This methods output a list of all variants for the given gene. The **gene** keyword should be replaced by a valid HGNC symbol.
- http://bioinformatics.ua.pt/WAVe/variant/COL3A1/atom: COL3A1 data in Atom format.

→ UniversalAccess: *http://bioinformatics.ua.pt/WAVe/gene/<gene>/<key>:<identifier>*

- WAVe's gene data feeds provide access to all data collected in WAVe. To access these data, users are redirected to WAVe, and the external resource is automatically loaded on arrival. This way, when WAVe's API is used the application always sends users to the original WAVe resource context. This method can also be used to build custom URLs to load specific identifier within WAVe's interface. The **gene** keyword should be replaced by a valid HGNC symbol, the **key** keyword must be replaced with a valid WAVe relationship keyword (the **ua** property in the configuration file) and a valid **identifier**. If the **key** parameter is not valid, WAVe displays the default gene page. Alas, it is impossible to track invalid **identifier** values when the key is correct, and this results in an error in the external resource page.
- http://bioinformatics.ua.pt/WAVe/gene/COL3A1/omim:130020: loads the OMIM page for 130020 (Ehlers-Danlos Syndrome, Type III) within COL3A1 workspace.

The most successful scenario for WAVe's API use is its inclusion in the GEN2PHEN project Knowledge Centre [189]. At this portal, when searching for genes, WAVe's API is contacted in realtime to retrieve information regarding available gene relationships. For instance, the GEN2PHEN Knowledge Centre displays links from WAVe's Locus, Publication, Disease, Genome and Gene Ontology nodes[36].

5.3 Discussion

5.3.1 Service Composition for Integration in Bioinformatics

An assessment of the bioinformatics research field from a computational science standpoint reveals the lack of integration infrastructures as a common denominator for various bioinformatics challenges. The fact that data are heterogeneous, scattered and divided is true whether the data are coming from sequencing hardware, like in most bioinformatics research laboratories, or from a distributed collection of repositories, such as in WAVe. Whatever the case, new strategies must be devised to reduce the complexity of publishing acquired data for exploitation in external systems.

In modern integration architectures, the boundaries between the studied data integration strategies are squandered, resulting in implementations combining the features that better suit the problems at hand. At this stage, factors such as scalability, flexibility, efficiency and performance are weighted against the constraints of adopting a unique approach. This hybrid integration strategy defined WAVe's architecture, data model and implementation.

DynamicFlow started the assessment of the best methods to describe external services for use in a composition scenario. This pursuit is continued in WAVe, with a resource description schema to enable the accurate integration of data from external sources. The technological contributions detailed in this chapter effectively bridge the gap from data and services in bioinformatics. Nevertheless, with them as also arisen the need for a more streamlined integration and interoperability development workflow, an endeavor further discussed in the next chapter.

5.3.2 A Unique Resource for Human Variome Data

Establishing relations between the genotype and the phenotype provides a sound basis for significant advances in individualized healthcare. Consequently, this domain is the subject

[36] http://www.gen2phen.org/gene/col3a1

of several large-scale research projects, each proposing distinct strategies to obtain new knowledge regarding the human variome.

From these projects arises the challenge of how to integrate human variome datasets from miscellaneous locus-specific databases and how to enrich available data with meaningful relationships to the most relevant life sciences resources. Hence, a new streamlined integration solution, sustained by a holistic and lightweight architecture, was introduced to complete a comprehensive set of requirements. The designed algorithms were then put together to build the WAVe platform, where the true value of locus-specific databases is at researchers' fingertips through four key facets:

- → The lightweight **integration** engine collects a core genetics dataset, gathered from a myriad of locus-specific databases, which is connected to several external resources, making the data richer and more meaningful.
- → The **extensibility** of WAVe's data model enables the easy addition of new connections to external resources [10]. Updating the integrated relationship set is as simple as configuring a couple properties: the build engine will take care of the actual import process.
- → **Interoperability** is not neglected in WAVe. The REST API allows any developer to get genetic variation datasets or rich gene data through simple and direct methods. The GEN2PHEN project Knowledge Centre already uses these APIs to load gene data in real-time.
- → WAVe's innovative **interface**, with the gene navigation tree, **Gene Mesh** and **Live View**, streamlines the exploration of integrated data and wraps external applications within WAVe, thereby establishing a set of connections between disparate systems that would not be possible otherwise.

WAVe delivers a unique perspective over the human variome research domain, providing rich integrated access to genetic variation datasets through an agile workspace publicly available online at http://bioinformatics.ua.pt/WAVe/ [11].

6. COEUS: AN APPLICATION FRAMEWORK FOR ENHANCED SERVICE COMPOSITION

"All truths are easy to understand once they are discovered; the point is to discover them."
—Galileo Galilei

The Semantic Web has provided bioinformatics developers with better paradigms, standards and technologies to solve common problems such as data heterogeneity, diversity or distribution. The Bio2RDF warehouse [55] or the Biocatalogue library [73], amongst other systems, have shown how valuable semantic web technologies can be for the general life sciences software field [190]. However, semantic web's potential is still out of reach of the bioinformatics developers' community. There is a clear absence of tools to enhance the migration of existing platforms to new environments, to ease the development of information systems from scratch or to disrupt with past strategies by deploying fully interoperable software.

Hence the introduction of COEUS, a framework to tackle these challenges by empowering developers with a "Semantic Web in a box" software stack, and ensuing a more agile development workflow for new semantic web systems [14]. The COEUS open-source project is available at http://bioinformatics.ua.pt/coeus/.

This chapter discusses the devised strategies and their implementation, leading to COEUS development. Starting with the demand for more modular and dynamic software packages in the life sciences, we move on to a brief assessment of semantic web use in bioinformatics, highlighting the opportunities for a new kind of application development strategy that can empower the next generation of bioinformatics software.

6.1 Dynamic Software Infrastructures for Life Sciences

6.1.1 Reusable Assets

The cornerstone of current software development is the idea of "reusing instead of rewriting". This rather basic proposition is applied not only to the construction of data models, where defining new schemas or entire structures is a complex research practice, but also to the set of programming toolkits being used. Regarding the latter, the vast number of applications, libraries, services or packages, makes it very easy for developers to find a solution to an implementation problem. Even when they end up implementing the desired feature from scratch, they do so acknowledging the already existing algorithms, their limitations and their strengths.

Common modeling, service access, knowledge acquisition or data exploitation problems have been solved before. Associated with the facility in finding existing solutions, developers are now endowed with tools to quickly integrate miscellaneous libraries in their projects, such as Maven[37], Node.JS NPM[38] or Ruby's Gems[39]. Hence, the software development process is streamlined to a three-stage identify-assess-reuse practice.

As stated in section 3.1, this is leveraging the use of rapid application development frameworks. Likewise, the bioinformatics field is also becoming aware of these quicker application deployment strategies, and new toolkits are starting to emerge.

Evaluating Rapid Application Development in Bioinformatics

Rapid Application Development (RAD) is a strategy for generating entire application, including databases, code and services, from a set of configuration files. This permits launching new tools much faster than otherwise, thus reducing the "time-to-market" cost. When assessing RAD strategies, the major conclusion is the traditional poor component availability. Available frameworks either generate one or two good components (going feature-deep in each one) or generate multiple components with basic functionality (going for a wider coverage breadth). Nevertheless, these frameworks permit creating complete application stubs in no time, making them suitable for initial prototypes or low-end solutions.

[37] http://maven.apache.org/
[38] http://npmjs.org/
[39] http://rubygems.org/

The Molgenis framework is a “generic, open source, software toolkit” to quickly produce bioinformatics user-friendly and scalable software [88, 191, 192]. This toolkit provides developers with a simple modelling language to design data structures and user interfaces. From two valid configuration files, Molgenis’ generator creates a "feature-rich, ready-to-use web application including database, user interfaces, exchange formats, and scriptable interfaces".

The automatic code generation tools are one of Molgenis’ highlights. They reduce the amount of code one has to write by hand. The template-based nature of available methods leverages a straightforward generation process, easing the transformation from the configuration file to SQL, Java, R or HTML code.

Molgenis’ use has been growing over the last few years. Biomedical applications for miscellaneous areas, including genome-wide association studies, proteomics, biobanking or next-generation sequencing, have already been launched using this toolkit. Bioinformaticians usually seek Molgenis’ great adaptability. This allows them to generate entire application structures much faster when the resulting skeleton can be optimized or to iteratively generate solutions until the final system is ready.

ProteoWizard, BioJava and AIBench are some of Molgenis competitors. ProteoWizard is a C++ framework very similar to Molgenis in the sense that it provides a comprehensive set of features to speed up the development of applications requiring some kind of proteomics data manipulation [193]. As the name states, BioJava is a set of libraries that can be used in any Java application project and that reduce the complexity of dealing with biological data [83]. This widely used package facilitates reading and parsing data, performing simple statistical and analytical tasks and accessing common bioinformatics features such as sequence alignment or protein structure exploration. At last, AIBench was initially built as a rapid application development framework for data mining, but its use is being extended to biomedicine [194]. Also in Java, AIBench enables code annotations, scripting and custom plugins to be included in a predefined application skeleton, reducing the huge effort of implementing desktop application interfaces from scratch.

Combining rapid service development with semantic technologies is SADI’s framework goal, which promotes guidelines and ontologies to exploit the composition of semantic web services, through a straightforward strategy [195, 196]. Input and output interfaces accept and expose data in RDF format: service data are OWL class instances. Inward data are enriched with new relationships until they match the desired output, they are then sent as the service reply, streamlining the web service dataflow. SADI includes patterns for describing the service interfaces and enables the creation of client applications with "strikingly rich semantic behaviours". Henceforth, it is clear that rapid application development strategies must be mixed with the semantic web paradigm to deploy richer bioinformatics application frameworks.

6.1.2 Towards a Semantics-enabled Architecture

Research from Slater *et al.* [197] and Kozhenkov *et al.* [198], among others, analyses current software development strategies, concluding that there is a clear need for new approaches adopting distinct ideals and based on a different set of skills. Like next-generation sequencing hardware improves genetic reads in a multitude of ways, the Semantic Web may be seen as a next-generation software development paradigm, capable of breeding a new wave of biomedical software solutions. However, despite its growing momentum, semantic web strategies are still subject of a slow adoption process.

Taking in account the need for novel bioinformatics software with improved integration and interoperability features [199], the use of semantic web technologies to tackle innate life sciences challenges will permit that entirely different computational systems exchange and accurately interpret knowledge. With an ever-increasing amount of data, produced in both novel software and hardware platforms, and a prolific research community constantly demanding best-of-breed tools, this field is evolving exponentially and reaching user types far beyond the traditional wet-lab biologist [200-202]. Semantic knowledge discovery, reasoning and inference are now a part of state-of-the-art research.

Despite the key role that bioinformatics software and hardware developments have played over the last three decades, the life sciences technological ecosystem is still fragmented and characterized by immeasurable entropy. The majority of data are scattered through closed independent systems, disregarding any good-practice for integration and interoperability features. Furthermore, even in notable state-of-the-art tools, the overwhelming scale and complexity of collected data and features generates an information overload, making it impossible for researchers to grasp any deep insights from available knowledge [203, 204].

Interoperable bioscience data are essential to keep up with the bioinformatics evolution momentum and extract the added value from the vast swathes of digital life sciences data [205]. This demands new strategies for getting the data out of primitive systems, using independent formats and non-standard terminologies, into a state-of-the-art open knowledge federation environment [13].

Furthermore, reusable data and reusable components are key for reproducible research and easily accessible knowledge. Making new systems part of a bigger network, such as the Linked Open Data cloud, will ultimately result in better access to data, promoting research collaboration and further increasing community buy-in.

To overcome these challenges we envisaged a new application development paradigm that boosts the integration of distributed and heterogeneous data and promotes interoperability through multiple application programming interfaces. Tying all this with semantic web developments results in a powerful methodology for improving existing biomedical software and streamlines the deployment workflow.

An Architecture for Knowledge Federation

Combining biomedical software engineering with semantic web ideals, we can pinpoint two broad and distinct approaches for enriching existing datasets with integrated connections amongst resources [206]. On the one hand, there are strategies based on data warehousing techniques, centralizing content from heterogeneous resources. On the other hand, there are solutions involving integrated access to distributed data sources, federating available content through a middleware layer. Both approaches are shown in Figure 6-1.

In opposition to warehousing, federation strategies acknowledge the distinct setup of each specialized instance. The integration of distributed resources requires some kind of middleware, a federation layer, to connect data available in each federated instance. Once this layer is deployed, data access becomes transparent. Even though performance may be poorer than in warehousing solutions, federation strategies can easily scale to accommodate distinct ontologies, regular data updates in each independent node and long-term improvements. Federation is hidden from end-users as they can access data in the same way as with warehousing repositories. Moreover, the federation layer handles query distribution and deals directly with each repository native API.

Furthermore, federation is innate to Semantic Web technologies and fits well within the biomedical applications domain [207]. The SPARQL specification was designed from scratch to ease this process. Publishing data through SPARQL endpoint enables access to data in more advanced ways than traditional SQL. Not only does this permit development of general federation tools, but it also promotes the creation of more complex software frameworks, sustained by native Semantic Web federation [12].

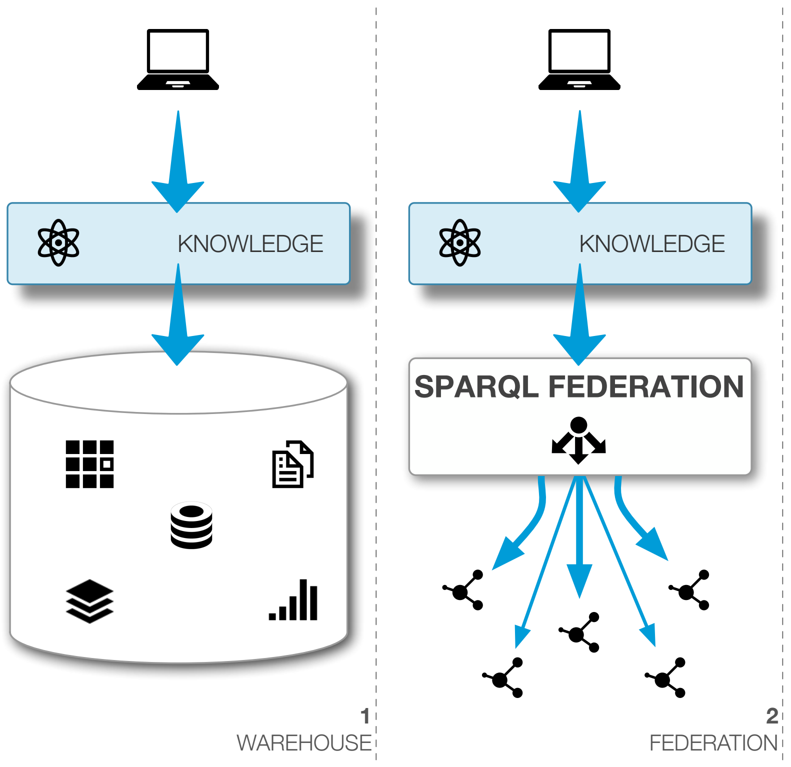


Figure 6-1. 1) Warehouse integration strategy, multiple resources are replicated in a central warehouse for prompt access to knowledge. 2) Federation strategy based on SPARQL endpoints providing direct access to each resource.

6.1.3 Semantic Web State of the Art in Bioinformatics

The semantic web migration process, moving systems from flat-file or relational environments to semantic infrastructures, has been the subject of extensive research [208-210]. The major emphasis is given to the development of translation languages, enabling the mapping from relational connections to the semantic web graph. On the one hand, basic languages map tables and columns to a new model following a proposed ontology. On the other hand, more innovative systems permit the extension of existing data connections, enriching their meaning and expressiveness. In this topic, two approaches are common. Some mappings are dedicated to forming new triple sets from existing relational databases whereas other languages enable publishing semantic views over relational data.

These languages are complemented with translation applications, using the newly mapped model to provide a semantic data version. Triplify [211], Virtuoso[40] and D2R server [212] have managed to employ new techniques that allow for semantic views and provide

[40] http://virtuoso.openlinksw.com/

access to existing relational data. Instances with DBLP[41], SIDER[42], DrugBank[43], DailyMed[44] and Diseasome[45] data were created using D2R, enabling SPARQL data integration endpoints.

Despite these advances in migration technology, the resulting systems are just a semantic version of pre-existing relational data. Thus, there is a lack of data insertion and triplification features, which are challenging tasks being backed by large-scale research projects.

Bio2RDF or DBPedia collect a gigantic amount of data in outsized triplestores. With the same decision-support goals as traditional warehouse systems, these applications adopt advanced extract-transform-load techniques to triplify existing data into a semantic format, storing them in triplestores. DBPedia offers a triplified Wikipedia version, containing its entire dataset, along with multilanguage support and category organization. Bio2RDF's focused biology environment enables it to be a remarkable life sciences semantic database, collecting data pointers from a wide array of domains, from genes to proteins up to pathways and publications.

Despite these large semantic systems' quality, they are not fit for common niche fields. Whilst Bio2RDF diversity and size will expand its use to the level of systems like UniProt or BioMART, these features also make it less suitable for smaller and restricted environments such as specific gene, disease or model organism information systems. Even if new software includes connections to Bio2RDF data, the system's core will be composed of small datasets and other precise information bits gathered from external databases or wet-lab file systems.

S3DB proposes a new data management model for integrating biomedical knowledge capable of helping in miscellaneous niche environments [213]. S3DB provides developers with tools to construct their own ad-hoc Semantic Web applications, instead of beginning the development with an empty box. The proposed solutions for managing ontologies or locked data repositories make it adequate for closed environments. Data integration features are still very primitive, though. In these areas, it is imperative to provide mechanisms for importing data in various formats into the triple store, a process essential for obtaining enhanced collections of data.

[41] http://www4.wiwiss.fu-berlin.de/dblp/
[42] http://www4.wiwiss.fu-berlin.de/sider/
[43] http://www4.wiwiss.fu-berlin.de/drugbank/
[44] http://www4.wiwiss.fu-berlin.de/dailymed/
[45] http://www4.wiwiss.fu-berlin.de/diseasome/

6.1.4 Opportunities for Building a Semantic Web Framework

Evolving current applications to the semantic web ecosystem is a necessary leap in upcoming years. With the currently available tools, successful migrations are limited to a below-reasonable level. Moreover, developers must take in account the needs of future software: the integration and interoperability challenge must be tackled from the start. The combination of these factors with biomedical software requirements demands a new kind of application framework, thriving under the vast potential and opportunities brought about by semantic web technologies. The reasoning for developing COEUS is summarized next, in four overarching integration and interoperability opportunities.

→ As previously stated, the principles for **rapid application development** practices, already common in the general computer science field, are gaining traction within the bioinformatics community. With this methodology, the opportunity arises to promote the use of streamlined development packages, libraries and frameworks.

→ The adoption of **semantic web integration** strategies, based on advanced knowledge triplification procedures, is a vital opportunity to improve existing Extract-Transform-Load tasks in data warehousing. Easing the transition process from CSV files or relational databases into semantic web triplestores is the cornerstone for publishing knowledge online.

→ With data being generated at a very high throughput rate, connecting it and making it available is essential to fully explore and understand its inner wisdom. Hence, there is a clear opportunity to employ new **semantic interoperability** standards, like SPARQL or LinkedData, to enhance knowledge sharing, broadcasting, reasoning and inference.

→ **Federated** data networks will play a key role in the future dissemination of knowledge from any science field. The demand of more integrative and interoperable data leverages the opportunity to build new systems where these features are standard and available by default.

With COEUS we introduce a solution that embraces these opportunities, being able to scale and adapt to future unforeseen scenarios. The COEUS framework offers flexible schema mappings for data integration and future-proof interoperability, making it the ideal candidate for improving the complex process of developing new semantic web application ecosystems.

6.1.5 Requirements Analysis and Design Issues

Leveraging on the aforementioned opportunities to build a new semantic web-based environment, we conducted a careful analysis of the requirements behind such system.

These requirements are generically entailed in the following guidelines: (R1) enhanced rapid application development, (R2) suitable integrative ontology, (R3) semantic data management, (R4) flexible integration, (R5) customizable web applications, (R6) interoperability with software tools, (R7) federation architectures and (R8) open-source availability. Next, a lightweight overview of these requirements is introduced:

- **(R1) Enhanced rapid application development.** The COEUS framework must bring rapid application development in bioinformatics one step further. This should be particularly evident in the adoption of semantic web technologies and in the bioinformatics-driven platform design.
  - **(R1.1) Streamlined instance configuration.** The configuration of new COEUS instances must be streamlined to require a minimal set of instructions.
  - **(R1.2) Simple instance boot.** COEUS instance creation process must be simplified and the majority of tasks automated to enable the quick launch of new applications.
  - **(R1.3) Usable in any programming environment.** The resulting framework must make data available for any client-side development environment.
- **(R2) Suitable integrative ontology.** COEUS' development must include the design of a new integration ontology, tailored to the devised integration strategies.
  - **(R2.1) Rich resource description.** The description of integrated resources must be as rich as possible to allow for a flexible integration environment.
  - **(R2.2) Ontology-based data mappings.** COEUS' ontology for resource description must enable the mapping of non-semantic data into any ontology from any field.
- **(R3) Semantic data management.** COEUS must be supported by a semantic knowledge base, thus enabling data management through a semantic layer.

- **(R3.1) Triplestore knowledge base.** COEUS' knowledge base should be supported by a semantic triplestore, whether through in-memory, file-based on relational-based strategies.
- **(R3.2) Semantic data translation.** COEUS must enable the translation of data from existing non-semantic environments into its internal semantic knowledge base.
- **(R3.3) Knowledge reasoning and inference.** As an integral part of any semantic web system, features must be available in COEUS to permit the effective reasoning over acquired knowledge and the inference of new data relationships.

→ **(R4) Flexible integration.** Resource integration in COEUS must be a flexible process to allow the integration of data from distributed and heterogeneous sources.

- **(R4.1) Data loading from SQL, CSV, XML or SPARQL sources.** Automated integration of data from CSV or XML files, or from SQL or SPARQL query results is mandatory.
- **(R4.2) Extensible integration architecture.** In addition to (R4.1), COEUS must support the creation of custom integration plugins.

→ **(R5) Customizable web applications.** COEUS must empower the eased creation of client-side web applications, through normalized infrastructures.

- **(R5.1) Internal API.** An internal Java API must be available to enable the creation of client-side Java applications.
- **(R5.2) JavaScript API.** Modern web applications rely on advanced JavaScript interactions. Therefore, COEUS must also include a direct JavaScript interface to its knowledge base.

→ **(R6) Interoperability with software tools.** The COEUS framework must assure interoperability with any external system.

- **(R6.1) REST API.** A generic REST API must be made available to permit the use of data from COEUS' knowledge base within any external system.

→ **(R7) Federation architectures.** COEUS' setup must support the creation of intelligent knowledge networks through the federation of data collected in each instance.

- **(R7.1) SPARQL API.** A SPARQL endpoint must be accessible to allow direct queries to each COEUS instance knowledge base.
- **(R7.2) LinkedData API.** A view adopting the LinkedData principles must be publicly available.

→ **(R8) Open-source availability.** All developed COEUS components must be provided through open-source licensing schemes.

Table 6-1 summarizes the relationships between these requirements and the problems encountered during our investigative literature analysis.

Table 6-1. Summary of relationships between the defined requirements and the issues overviewed in the researched scientific literature.

	R1	R2	R3	R4	R5	R6	R7	R8
Swertz *et al.* [88]	●							●
Wilkinson *et al.* [195]	●			●		●		
Cannata *et al.* [190]	●	●	●			●	●	
Slater *et al.* [197]				●		●	●	
Kozhenkov *et al.* [198]			●	●				
Hepp *et al.* [199]					●	●		
Cannata *et al.* [200]		●	●					
Marx *et al.* [205]			●	●				●
Cheung *et al.* [207]						●	●	
Hazber *et al.* [210]			●	●				

6.2 COEUS: A Semantic Web Application Framework

Semantic Web tools enable translucent relationships amongst data. The semantic web itself is a truly intelligent data network, with rich connections allowing for a better understanding of available knowledge. However, despite the immense possibilities surrounding semantic web technologies, its adoption has been dimmer than anticipated. Whilst stakeholders from all domains acknowledge the benefits of having a fully semantic information system, the difficult transition from traditional flat-file or relational database supported systems to the semantic web is a challenging roadblock [214].

6.2.1 Framework Setup

COEUS' "Semantic Web in a box" strategy envisages the inclusion, in a single package, of all the tools required to launch a new Semantic Web based application. In addition to this, COEUS' setup must also account for a flexible and scalable deployment environment. Many of the architectural decisions observed when implementing this framework had these ideals in mind. Hence, various tools and platforms were evaluated in the search for the optimal combination of components and integration/interoperability strategies that could transform semantic web rapid application deployment.

To better explain COEUS' strategy we employ a naming strategy that adopts a gardening metaphor. A single COEUS instance is entitled as **Knowledge Seed**, or simply **seed**. In scenarios with multiple **seeds** deployed in a true application ecosystem, this federated structure is envisaged as a **Knowledge Garden**.

Knowledge Representation

As mentioned in chapter 3, data in the Semantic Web are stored in formal triple statements: Subject-Predicate-Object. These statements employ different vocabularies and languages to identify each component. We can make a simple analogy to basic sentences with a **subject**, a verb representing action or meaning - the **predicate**, and what relates to the **subject** - the **object**. One last thing to consider is that predicates relate to object or data properties. Figure 6-2 highlights this division in a common sentence matched to a single statement.

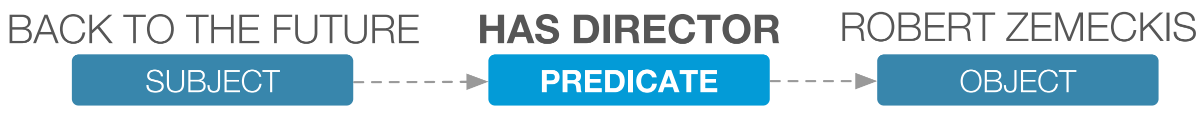


Figure 6-2. Sample triple statement, subject – predicate – object.

Taking in account the multitude of data models we can integrate within a single COEUS instance, the general semantic web knowledge representation strategy is more than fit. This allows us to map any content from CSV columns or SQL query results into a set o RDF statements.

For the knowledge storage framework component we identified and assessed various RDF management tools, as briefly covered in Table 6-2. The Jena framework is the most suitable alternative for COEUS' knowledge base. Its Java-based nature, easy integration with other tools and extensibility, make it ideal for use in a component-based framework. Jena's

API has basic support for reading and writing triple statements in Java in an in-memory or database-supported triplestore.

Table 6-2. Knowledge storage and representation technologies comparison.

FRAMEWORK	DESCRIPTION
Jena[46]	Widely used semantic web package for Java. Includes several features to easily deploy new applications, including support for SPARQL queries, RDF and OWL APIs, and inference. Provides multiple storage and reasoning mechanisms and also allows the integration of custom data processing mechanisms.
Sesame[47]	Widely used RDF framework and server. Includes support for SPARQL queries and an HTTP server interface. It is packaged with multiple storage and reasoning mechanisms and also allows the integration of custom mechanisms.
Virtuoso[48]	Widely used commercial solution for semantic web development. Includes a platform agnostic solution to access data through SPARQL queries, manage knowledge bases and integrate heterogeneous resources.
Redland[49]	Collection of RDF libraries for C, with bindings for various other languages. Provides RDF API, parsers, and query interfaces.
LinqToRDF	Semantic Web framework for .NET built on the Microsoft Language-Integrated Query (LINQ) Framework (language-independent query and data processing system).
OWL API[50]	OWL API and implementation for Java. Includes an OWL API that is built on the functional syntax of OWL 2 and contains a common interface for many other reasoners.

Components

The basic COEUS setup requires a Java application server (Tomcat is recommended) and a relational database (for the triplestore backend). All the other necessary components are included in COEUS package, further facilitating the creation of new systems from scratch.

Figure 6-3 highlights the component interactions in each standalone instance and Table 6-3 describes all used components and their purpose within the framework.

[46] http://jena.apache.org/
[47] http://www.openrdf.org
[48] http://virtuoso.openlinksw.com/
[49] http://librdf.org/
[50] http://owlapi.sourceforge.net/

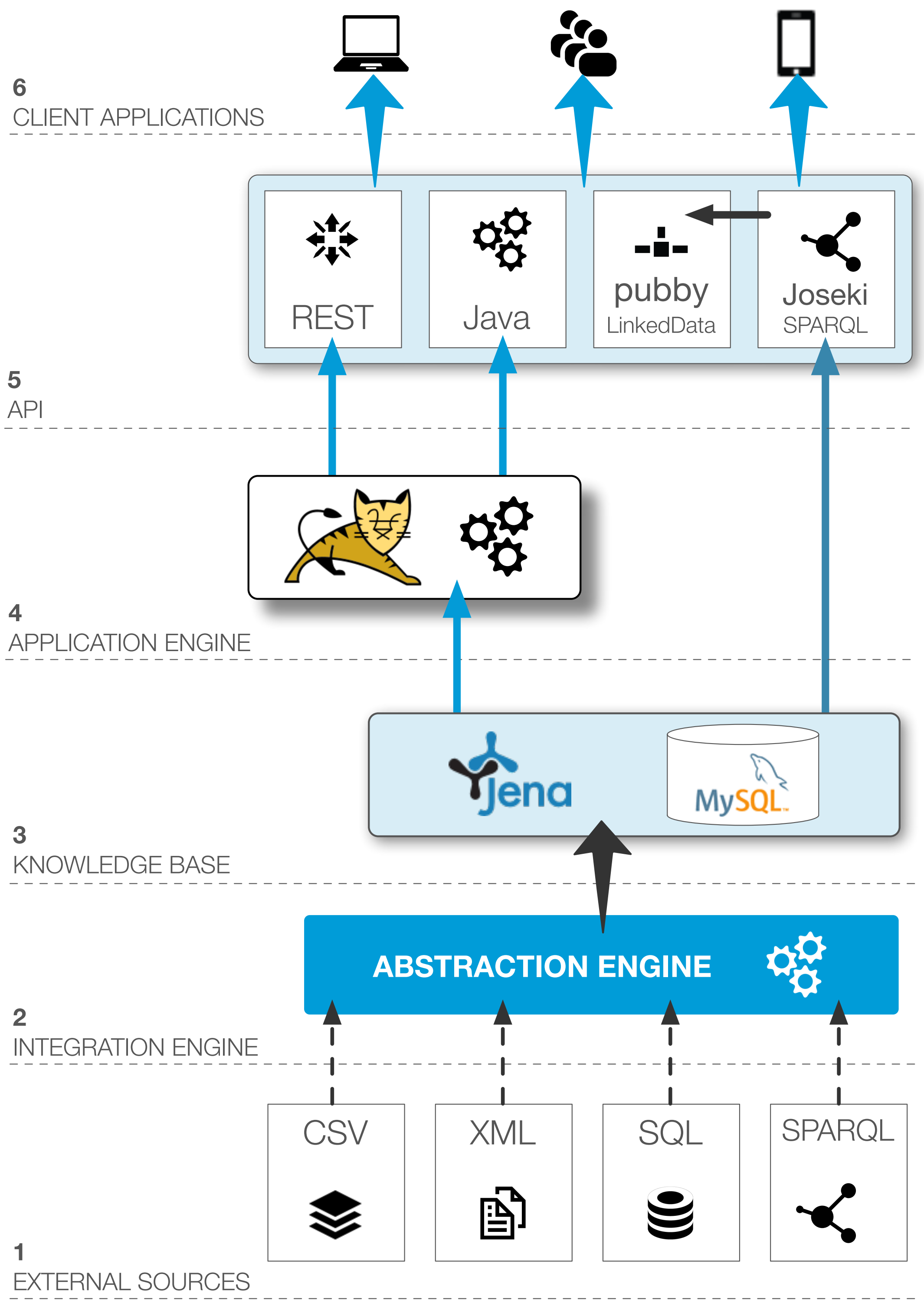


Figure 6-3. COEUS seed component interactions diagram. 1) External data are integrated from multiple sources using the available CSV, XML, SQL or SPARQL connectors. 2) The abstraction engine translates read data into the seed knowledge base. 3) COEUS internal triplestore is supported by Jena **with a MySQL relational database backend. 4) Data in the knowledge base are accessed through the application engine for the Java and REST APIs. 5) The SPARQL endpoint, provided by** Joseki**, allows direct access to the knowledge base and is used by** pubby **to enable the LinkedData views.**

Table 6-3. COEUS' framework libraries listing and descriptions.

LIBRARY	DESCRIPTION
Jena	Jena is used as the core semantic web package within COEUS, mediating input access to the knowledge base when building the triplestore and output access to the Java API.
Joseki[51]	Provides the SPARQL endpoint functionality.
Pubby[52]	Provides the LinkedData interface.
Sparql.js	JavaScript library to query remote SPARQL endpoints.
Tomcat	Java application server.
MySQL	Backend support to the Jena SDB triplestore.

Architectures

COEUS' application models and internal seed architecture can be combined in a single view, highlighting the knowledge flow from the data integration connectors to the interoperability API, detailed further in this chapter. The architecture for a single seed is show in Figure 6-4.

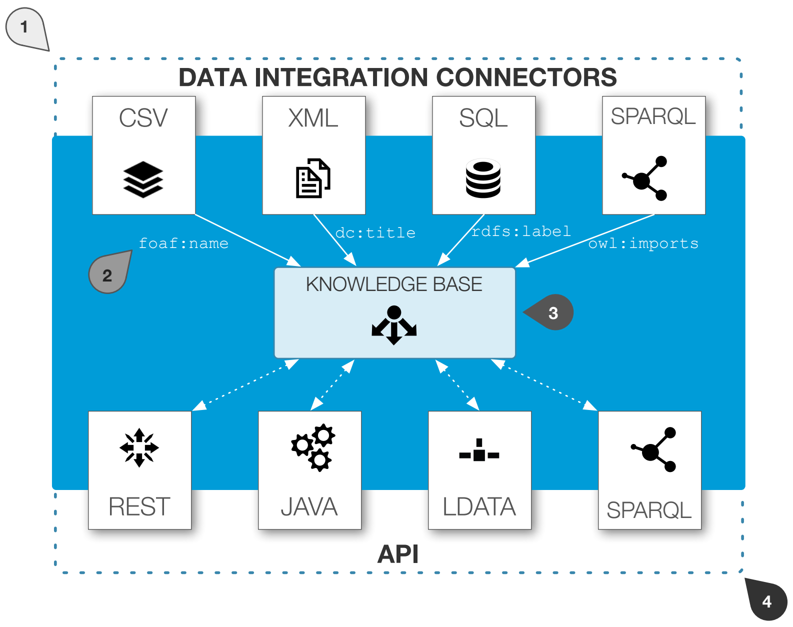


Figure 6-4. COEUS architecture for a single seed. 1) Data integration connectors for CSV, XML, SQL and SPARQL enable the triplification of data into each seed's semantic storage. 2) Data are selected from each external resource to match specific ontology predicates, generating a rich knowledge base. 3) COEUS central knowledge base includes the triplestore data repository and respective data access methods. 4) Acquired data are available through COEUS API, using Java methods, REST services, a SPARQL endpoint and the LinkedData view.

[51] http://www.joseki.org/

[52] http://www4.wiwiss.fu-berlin.de/pubby/

To further increase COEUS' innate flexibility, multiple seeds can be combined in a knowledge garden, providing a virtual holistic access layer to collected knowledge, regardless of its original location (Figure 6-5).

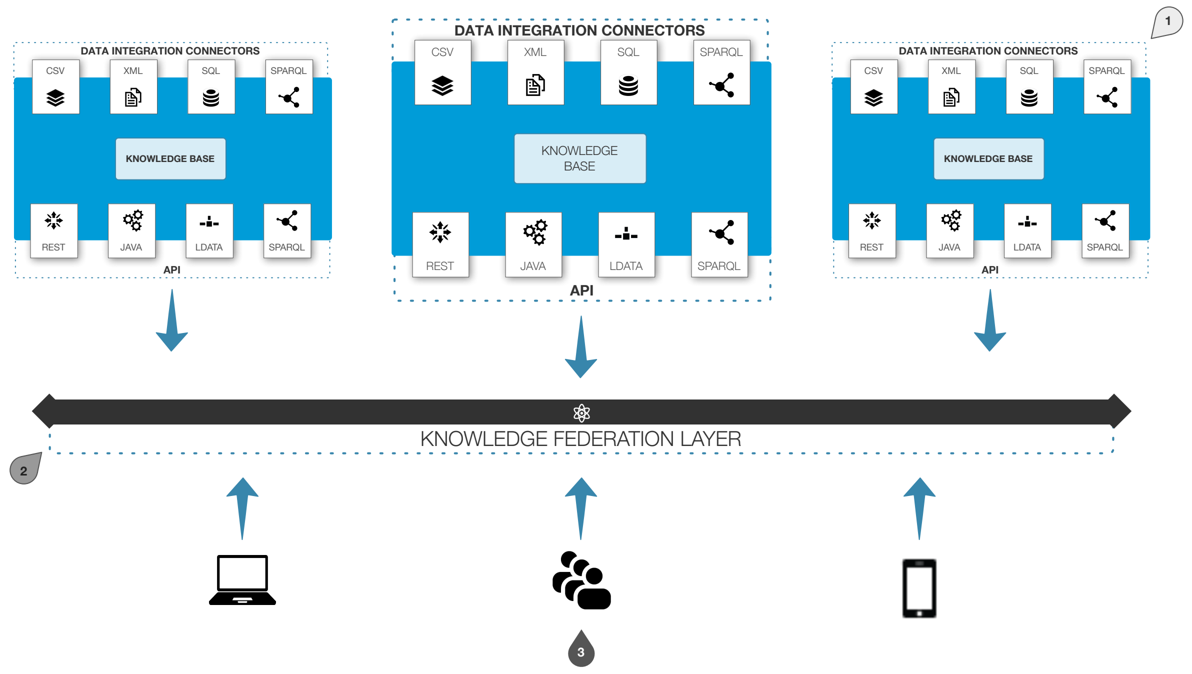


Figure 6-5. COEUS' garden architecture overview. 1) With one or more seeds in place, the COEUS platform enables a knowledge federation layer. 2) The distributed knowledge federation layer is capable of answering specific research questions. 3) This distributed entry point enables the deployment of multiple applications to web, desktop or mobile environments.

Application Models

The COEUS framework can be used to deploy distinct application models (Figure 6-6). On the one hand, a seed can accommodate multiple end-user applications, on distinct devices for instance. This permits that a single centralized data source can be built to support any number of web, desktop or mobile applications. On the other hand, multiple specialized seeds can be connected to supply a single holistic application. In this strategy all seeds work independently, and can be seen as nodes in a data sources network, providing access to an overarching tool. Along with these opposite strategies, hybrid architectures are also possible, combining multiple applications and seeds in a distributed data and applications ecosystem.

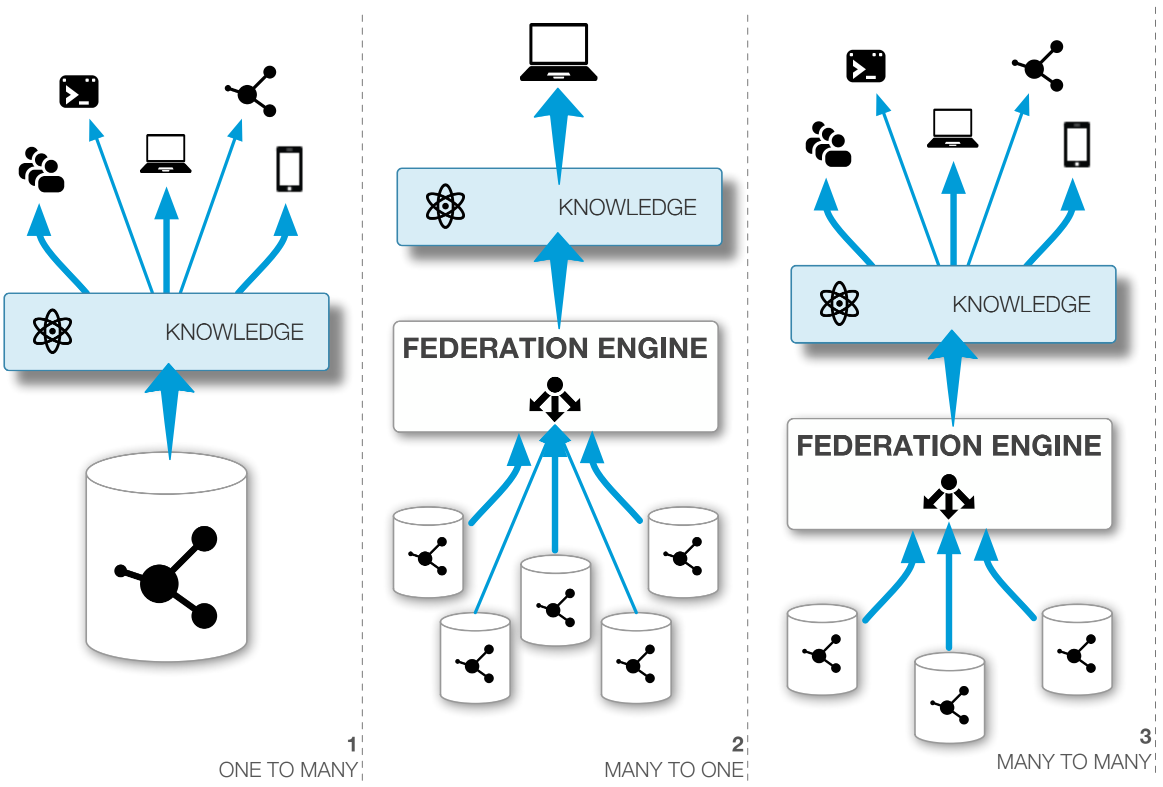


Figure 6-6. Basic COEUS application models. 1) One to many: one seed provides data to multiple client applications. 2) Many to one: using SPARQL federation multiple seeds are dynamically interlinked and accessed through a single application platform. 3) Many to many: hybrid model where multiple seeds are accessed by multiple client applications through a distributed federation layer.

Internal Ontology

To achieve the desired COEUS' scalability and flexibility, the basic platform model is organized in a tree-based structure. Data relationships are mapped to **Entity**-**Concept**-**Item** structures, which are connected to **Resources** and **Bridges**, supporting integration and exploration settings, respectively (Figure 6-7). This ontology is available online[53] and must be used in COEUS' configuration files. Describing COEUS entire ontology is out of this thesis scope. For further information regarding all classes, data and object properties or the entire structure, refer to COEUS' online documentation at http://bioinformatics.ua.pt/coeus/documentation/. A short description for each core class in the ontology follows.

→ A **Seed** is a single framework instance. In COEUS' model, **Seed** individuals are used to store a variety of application settings, such as component information, application descriptions, versioning or authors. **Seed** individuals are also connected to included entities through the **:includes** property (inverse: **:isIncludedIn**). This

[53] http://bioinformatics.ua.pt/coeus/ontology/coeus_1.0b.owl

permits access to all data available in the seed, providing an overarching entry point to the system information.

- **Entity** individuals match the general data terms. These are “umbrella” elements, grouping concepts with a common set of properties. For example, to gather proteomics information, the model has a “Protein” entity or for disease information there is also a general “Disease” entity. To better understand this organization, object-oriented structures, their inheritance and variable subtypes must be remembered.
- **Concept** individuals are area-specific terms, aggregating any number of items (the **:isConceptOf** property) and belonging to a unique entity (the **:hasEntity** property). Continuing the previous scenario, “UniProt”, “PDB” and “InterPro” databases are concepts within the “Protein” entity. Note that an **Entity** may have any number of concepts, but a **Concept** belongs to a single **Entity.**
- **Item** individuals are the basic terms, with no further granularity and representing unique identifiers from integrated datasets. In the above proteomics scenario, “P51587”, “P02461” are items under the “UniProt” concept, each matching a unique term from the original UniProt database. For the disease entity, the “104300” individual is a match for Alzheimer’s disease entry in the OMIM database concept. In the knowledge base, items can be associated to other items directly (predicate **:isAssociatedTo**) or through connections from their parent concepts. Entities are also connected to concepts, and these to items, making **Seed** individuals a central registry for COEUS’ seeds.
- **Resource** individuals are used to store external resource integration properties. The configuration is further specialized with **CSV**, **XML**, **SQL** and **SPARQL** classes, mapping precise dataset results to the application model, through direct concept relationships. In the proteomics scenario, a Resource individual contains information for the “UniProt” concept original data source, including its location and how to extract each item. This resource is connected to several **XML** individuals (predicate **:loadsFrom**), each containing an XPath query whose results will map to application model properties. With the **:hasResource** property, the framework knows exactly what resources are connected to each concept and, subsequently, how to load data for each independent concept, generating new items.

→ **Brigde** individuals are also mapped to concepts, storing concept visualization and exploration features. That is, bridges tell the system how concept items can be shown to users. This configuration permits any number of internal properties as long as they are understood by the final client application. This means that we can include parameters for advanced data visualizations, triggering web service calls or composing simple links. An example of the latter is a bridge for the "UniProt" concept, declaring a structure for building valid UniProt links, replacing *#replace#* in http://www.uniprot.org/uniprot/#replace# with individual item identifiers.

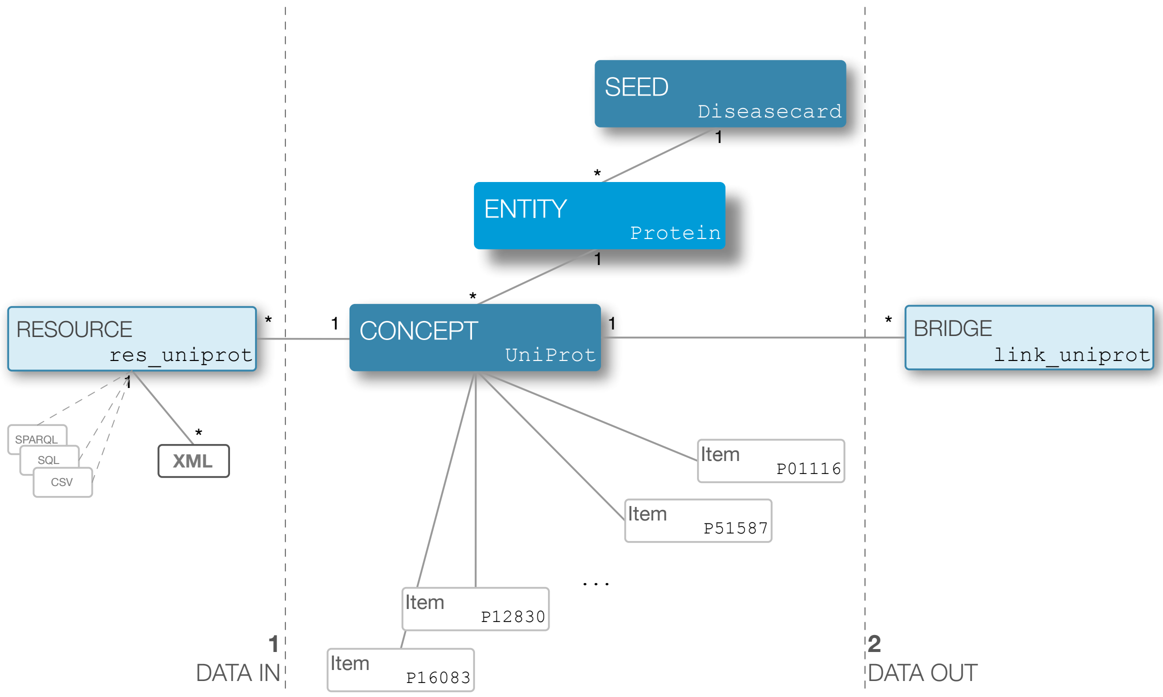


Figure 6-7. COEUS' ontology model for the internal tree-based structure highlighting relationships amongst the various individual classes. A seed can have multiple entities, and each entity can be related to one or more concepts. Concepts aggregate unique items and are connected to resource and bridge information. The data import process uses resources' properties (1) and custom methods can be defined to display data (2). Sample seed data for a "Diseasecard" seed is shown at each element, listing "UniProt" items belonging to a "Protein" entity.

One of the great advantages of using semantic web technologies is that any external ontology can be used to complement or extend COEUS' internal model. As long as new properties are understood by the seed applications, any number of properties can be added, mapping concepts or entities to existing ontologies or adding further properties to describe resources or bridges.

Data Flow

The COEUS framework gives developers total control over the data flow, from distributed repositories to the internal semantic knowledge base and from this to any end-user application. From a data input perspective, the goal behind this strategy is to provide developers with advanced methods to load precisely what they want, how they want it and from where they want it. Furthermore, on a data output perspective, we also want to provide enough flexibility for developers to build their own applications in any programming environment. To summarize, the inwards data flow establishes COEUS as a data integration platform (Figure 6-8) and the outwards data flow demonstrates its advanced interoperability features (Figure 6-9).

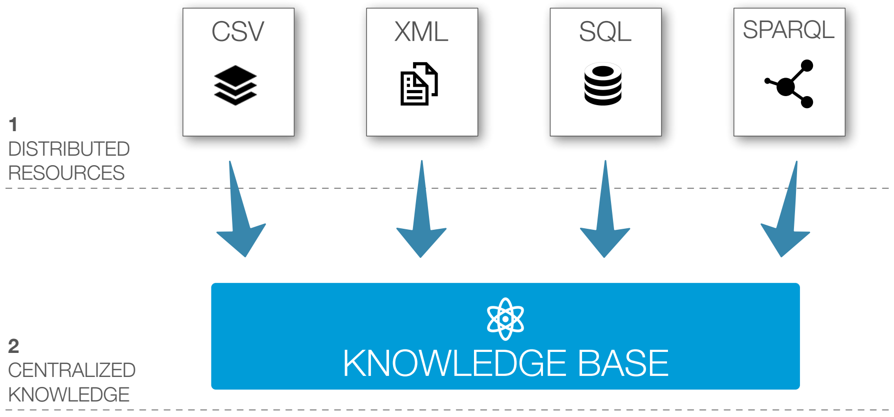


Figure 6-8. COEUS' inward data flow, from external distributed and heterogeneous resources (1) into a centralized knowledge base (2).

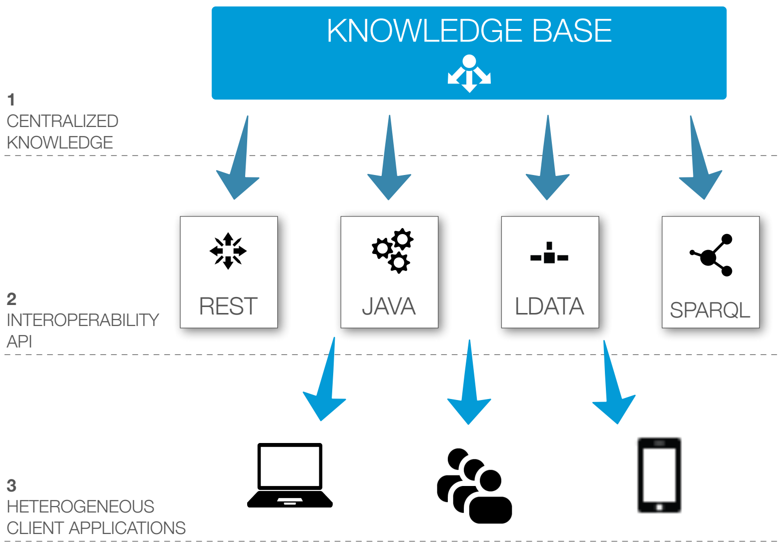


Figure 6-9. COEUS' outward data flow. Knowledge collected in the centralized knowledge base (1) is accessible through an interoperability API composed of four key interfaces: REST, Java, LinkedData and SPARQL (2). 3) The API enables the creation of miscellaneous client applications, target at multiple environments.

6.2.2 Extract-Transform-Load

Data integration is a perennial challenge in modern bioinformatics. As discussed in previous chapters, caveats such as resource distribution and heterogeneity transform integration into a demanding computer science challenge. One of COEUS' goals is to tackle this challenge. This framework provides features to facilitate the integration of heterogeneous data from distributed resources in an elegant fashion. Traditional warehousing techniques revolve around advanced algorithms for extracting data from a specific source, transforming it into the warehouse model and actually replicating the data in the new integrated dataset. In COEUS, this **Extract-Transform-Load** process is specialized to a semantic web environment, enhancing the inwards data flow from CSV, XML, SQL or SPARQL data to sets of triple statements.

Heterogeneity also appears in the distinct data models of each integrated resource. COEUS tackles this issue with a semantic web translation process. Due to COEUS roots, the internal knowledge base is model-agnostic, liberating integrated data from the restrictions of CSV tables or relational databases. Since all data are stored as triple sets, the limitations adjacent to foreign keys or table columns are replaced by meaningful relationships.

In most cases, the various properties stored in object-oriented models or XML structures can be re-engineered through the adoption of existing ontologies or the creation of new ones. As mentioned before, the usage of controlled ontologies augments the flexibility of internal data models, enabling the creation of tightly integrated datasets.

The physical and logical content heterogeneity issues impose the development of a generic data-loading tool. For simplicity purposes, a seed's configuration file includes the type of resource being loaded and the URI to access it. This way, all COEUS needs is an Internet connection to access REST or SOAP services, SQL databases or SPARQL endpoints. Furthermore, the URI naming scheme also permits the identification of local resources and SQL database connections can be made to a local host. This results in having the same structure in COEUS for importing local or remote data.

This immense amount of variables and configuration properties for integrating data lead to the appearance of the **connector** and **selector** concepts, explained further in this chapter.

Collecting Distributed Data

The initial problem that arises when building new warehouses or integrated datasets relates to the diversity of formats involved in the data import process. Whether we are accessing a REST web service or a MySQL database, most programming technologies allow configuring this access through a simple URI. For instance, a sample JDBC connection string to a MySQL database is

```
jdbc:mysql://thedbhost.com:3306/thedbname?user=thedbuser&password=thedbuserpwd
```

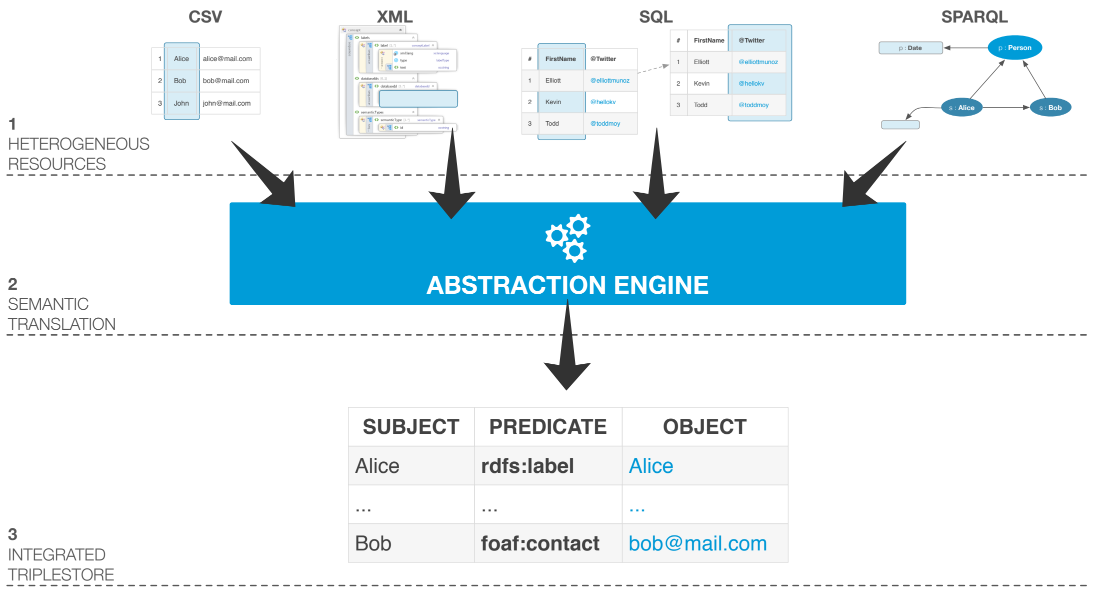
and a sample URL for accessing Twitter's API is

https://twitter.com/#!/search/%40term.

The similarities are clear and enable the simplification of the external resources configuration. All resources will have a URI property for instance.

If data format heterogeneity poses the initial threat for a linear data integration process, the data model heterogeneity further heightens it. We know from the start that data will come through in all sorts of formats and models. To overcome these caveats, COEUS adds an intermediate abstraction layer between the external resources and the internal knowledge base - Figure 6-10.

The idea behind this abstraction layer is to convert the data being integrated to a general model-independent format. In practice, the implemented method simply generates a network graph for each new item, mapping the configured predicates to the values from external resources. With this data abstraction, the triplification process can take place, enabling the generation of triple statements from the abstracted data model for further storage in COEUS' knowledge base.



SUBJECT	PREDICATE	OBJECT
Alice	rdfs:label	Alice
...	...	...
Bob	foaf:contact	bob@mail.com

Figure 6-10. COEUS' data abstraction process. 1) Data from external, distributed and heterogeneous resources is prepared for triplification. 2) COEUS connectors initiate the semantic translation using the abstraction engine. 3) Generate triples are stored in the seed knowledge base, a fully integrated triplestore.

Connectors and Selectors

The integration task consists in the acquisition of data from heterogeneous and distributed resources to populate a seed. This complex strategy required the construction of purpose-specific wrappers. These methods access external resources and process data, using the **connectors**, based on a set of configuration properties, the **selectors**.

Selectors are property sets defining the data location in a specific resource and what predicate will be added to the knowledge base during the integration process. Connectors control these particular data mappings: independent and generic modules to load information from external resources in CSV, XML, SQL or SPARQL formats. They possess a common set of configuration properties defining the data type, where the data are located, the relationships to existing data, and other module-specific definitions. This information is stored in the seed configuration files, exemplified in the following chapter. For instance, XML module configuration must include the original data source address and a collection of selector properties, XPath queries, which will be performed against the read XML, corresponding to the data being mapped. Likewise, SQL query column names, CSV column numbers, or SPARQL variables are used as selectors in their respective modules.

The data loading process uses connectors to initiate a data triplification process. Data are enriched through the dynamic generation of new triples based on specified configuration properties. With this Semantic Web-based Extract-Transform-Load we are augmenting the scope of data in one-dimensional CSV files or bi-dimensional SQL tables to a multi-dimensional triplestore.

The richness of this triplification process resides on connector's flexibility. The selectors within a given connector allows us to match any content into our semantic graph using a primary key for the subject, any property mapped from the seed ontology as predicate, and selection results as objects. These are then used to generate each new item map on-the-fly, which is then converted into a set of statements and inserted into the knowledge base.

Triplifying Content

The triplification process highlighted in Figure 6-11 may proceed in two modes: explicit and implicit. Explicit translations are required when data are read from CSV, XML or SQL resources. As data does not possess any related semantics (only columns or objects), explicit descriptions for the new predicates need to be set up. Since we can map data in XML, CSV or SQL to any predicate in any ontology and to more than one predicate at once, we can expand the meaning of non-semantic data by explicitly declaring it as the object of a specific statement.

SPARQL resources provide implicit semantics; data are already in the same semantic format (triple statements) as required by the storage engine. While loading data with the SPARQL connector, the selector can match data into new predicates or the original predicate. This way, COEUS can simply replicate triples or expand them to richer entities.

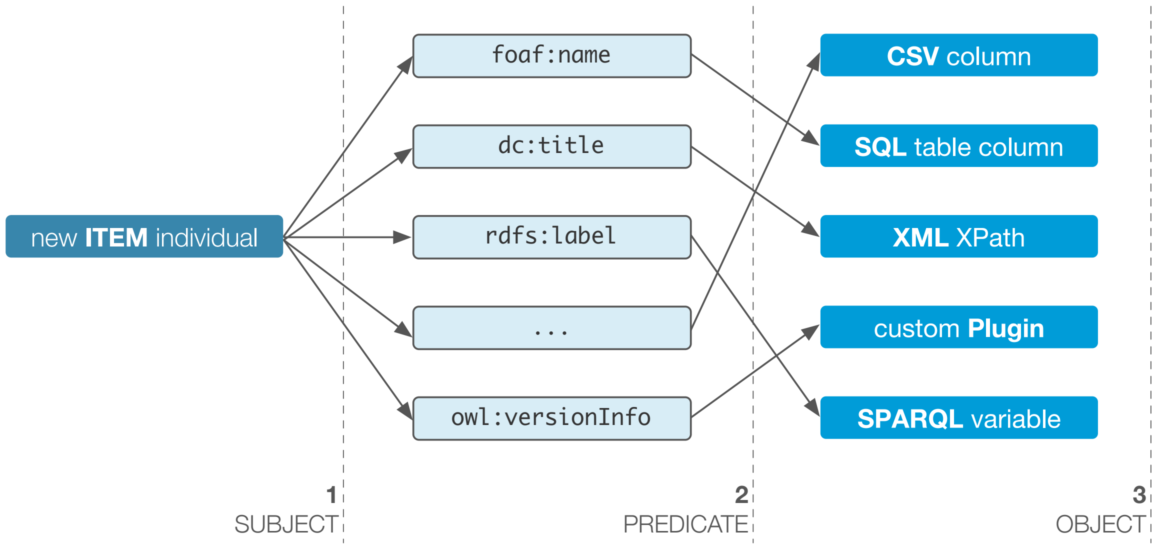


Figure 6-11. Triplification process overview. 1) Subjects, new Item individuals, are generated at runtime. 2) Predicates are read from the configuration file to match any predicate from any ontology. 3) Objects read from the external heterogeneous resources finish the triple statement.

In addition to these two triplification modes, data integration can also be performed using three distinct approaches according to the seed needs or to how data are provided by each service. These methods are defined by the **:method** property in a resource configuration. *Cache* is the default method and enables standard data loadings from external resources, generating new items and triplifying all data. The *complete* method adds new triples to items already in the seed triplestore. At last, the *map* integration method enables the creation of custom direct relationships amongst individuals. With this, we can create entire sets of new mappings amongst items after the data are loaded. Both the *map* and *complete* integration methods use the **:extends** configuration predicate from COEUS' internal ontology to define the Concept whose individual item list will be enriched.

With COEUS' triplification strategy, the data integration approach is abstracted from the data itself. Since there is ground for one or more common underlying ontologies, new axioms can be established disregarding traditional software constraints. Data can be collected and connected using distinct methods and miscellaneous import formats. This is ideal for optimizing all kinds of new data-powered applications, namely on the life sciences field, where heterogeneous data models with limited relationships are common.

Configuring a New Seed

The seed configuration controls the entire instance operability. At the moment, three separate files are used to set up application properties, the application model and resource integration properties.

- The ***config.js*** file stores volatile application properties. These include the application name, version, short description, deployment environment and the list of ontologies used in the seed. Using a JSON object for the configuration permits faster reads when compared to XML, while maintaining a good object-oriented structure, in comparison to simple properties files.
- The information system ontology. In most scenarios, the "reuse instead of rewrite" principle does not suffice for the entire application ecosystem. As such, COEUS allows the creation of custom ontologies to use in one or various seeds. Developers are able to organize their own applications models, taking full advantage of RDF/OWL's modelling flexibility.
- The application setup file includes the data integration and exploration configurations. In this file we define the individuals for each class, configuring entities, concepts, bridges and resources. Summarily, content in this file is used to guide the entire framework instance setup, from the handling of external resources in the connectors to the labelling rules for each Item individual.

COEUS' future developments include the addition of a user-friendly GUI to configure new seeds. In the meanwhile, and considering the setup files OWL/RDF nature, relying on Protégé is advisable to ease the configuration process. In this widely used ontology-modelling tool, the configuration can be written, tested and visually organized.

The amount and variety of configuration options is even greater than WAVe's. Hence, the best option is to look at the examples and online documentation at http://bioinformatics.ua.pt/coeus/documentation/. For mere descriptive purposes, a subsection from a real configuration file is included next. This code sample configures the loading of known human genes into a seed. This list, mentioned in WAVe's integration description, is maintained by the HUGO Gene Nomenclature Committee, which provides a REST service for getting the list in CSV format. This resource loads the list into our seed, populating a *HGNC* **Concept** under the *Gene* **Entity**. The **resource_HGNC** individual is configured to load the data from the selected **:endpoint** object, send it for processing using the CSV connector, property **dc:publisher**, and map the results from two **:CSV** individuals, property **:loadsFrom**. In these, the **:query** predicate defines which column object will map to the predicates listed in **:property**. Hence, in the **csv_HGNC_id** individual, data obtained from column *0* of the *HGNC* CSV file, property **:query**, will be mapped into two triple statements with the same subject, the **dc:source** and **dc:identifier** predicates and with the same object, property **:query**.

```
# HGNC Resource configuration
:resource_HGNC rdf:type :Resource , owl:NamedIndividual ;
   rdfs:label "resource_hgnc"^^xsd:string ;
   dc:title "HGNC"^^xsd:string ;
   :method "cache"^^xsd:string ;
   dc:publisher "csv"^^xsd:string ;
   :endpoint "http://www.genenames.org/cgi-
   bin/hgnc_downloads.cgi?title=HGNC+output+data
&hgnc_dbtag=onlevel=pri&=on&order_by=gd_app_s
ym_sort&limit=&format=text&.cgifields=&.cgifi
elds=level&.cgifields=chr&.cgifields=status&.
cgifields=hgnc_dbtag&&where=&status=Approved&
status_opt=1&submit=submit&col=gd_hgnc_id&col
=gd_app_sym&col=gd_app_name&col=gd_status&col
=gd_prev_sym&col=gd_aliases&col=gd_pub_chrom_
map&col=gd_pub_acc_ids&col=gd_pub_refseq_ids" ^^xsd:string ;
   :extends :concept_HGNC ;
   :isResourceOf :concept_HGNC ;
   :hasKey :csv_HGNC_id ;
   :loadsFrom :csv_HGNC_id,:csv_HGNC_symbol.

# HGNC CSV connector configuration for HGNC identifier
:csv_HGNC_id rdf:type :CSV , owl:NamedIndividual ;
   rdfs:label "csv_hgnc_id"^^xsd:string ;
   :query "0"^^xsd:string ;
   dc:title "HGNC_id"^^xsd:string ;
   :property "dc:source|dc:identifier"^^xsd:string ;
   :loadsFor :resource_HGNC ;
   :isKeyOf :resource_HGNC .

# HGNC CSV connector configuration for HGNC name
:csv_HGNC_name rdf:type :CSV , owl:NamedIndividual ;
   rdfs:label "csv_hgnc_name"^^xsd:string ;
   :query "2"^^xsd:string ;
   dc:title "HGNC_name"^^xsd:string ;
   :property "rdfs:comment|dc:description"^^xsd:string ;
   :loadsFor :resource_HGNC .
```

For improved dependency management, seed configurations are organized as graphs. That is, developers can implement dependencies amongst resources, enabling the loading of data based on previously collected individuals, selected with the **:extends** property. This allows for the creation of advanced data integration workflows, combining multiple concepts, thus enabling the aggregation of millions of triples in the seed's knowledge base.

6.2.3 APIs

COEUS tackles the lack of interoperability in existing life sciences information systems. Drawbacks such as poor web service availability, complex and closed data models, or vendor-specific formats are common in bioinformatics. To overcome these clear issues in semantic interoperability, COEUS includes, by default, a comprehensive API to explore collected data.

Available methods were developed with two goals in mind. On the one hand, data must be easily available for the creation of new applications within a COEUS seed. On the other hand, integrated data must also be published externally, making it available for any external system. Hence, COEUS' API is organized in two sections: internal and external, despite their natural promiscuity.

The internal API comprises the Java methods and Javascript libraries. The former provides an abstraction over Jena's basic data access functions and are adequate for scenarios where the seed client-side application is also being developed in Java. These methods permit data access in both ways, allowing for streamlined data access and traditional data insertions. The available Javascript library simplifies the process of accessing a SPARQL endpoint using Javascript. Combining this tool with modern user interface frameworks (such as jQuery) makes it very easy to query a seed's knowledge base and use the response data in the application. Consequently, developers can create highly interactive user interfaces, in any development framework. Moreover, custom endpoints can be configured in the JavaScript library to access data from external SPARQL endpoints. This enables the creation of modern semantic data mashups on the client-side.

The external API comprises a set of REST services, a SPARQL endpoint and a LinkedData viewer. The available REST services allow accessing content in multiple formats (CSV, JSON, RDF/XML or HTML). Likewise, the SPARQL endpoint is open for querying. With this endpoint, any query can be performed to exploit the wealth of integrated data. At last, the LinkedData perspective makes the knowledge base content available to any LinkedData browser, delivering an advanced structured interface to access data.

Java and Jena

While Jena provides a developer-friendly API for adding and retrieving data in Java, COEUS includes an additional set of methods to ease these tasks and facilitate data access.

COEUS Java API is an abstraction over Jena's internal methods, providing a more direct way to access COEUS data structures. Hence, accessing items, concepts or entities, or adding new statements actions are more straightforward. Next, there are the signatures for functions to add new statements and retrieving the result set of a SPARQL query. More examples and full documentation can be found online in the Java documentation at http://bioinformatics.ua.pt/coeus/javadoc/.

REST

COEUS' RESTful services API provides a set of methods to access data in the knowledge base through simple GET requests. REST services are currently the most widely used strategy for systems interoperability. Modern service-oriented architectures rely on these types of services due to their flexibility in regard of formats and operation types. The trade-off between having a more standardized (though constrained) services platform using SOAP and a more "open" alternative with REST was acceptable for COEUS, promptly pushing the latter as the only viable solution for supported services in COEUS.

Furthermore, in spite of the relatively low number of REST services available by default, more services can be easily added through the combination of internal Java methods with Stripes' powerful URL binding mechanisms. The Stripes framework has a very light learning curve, enabling the addition of new actions and services an easy job even for non-experienced Java web developers.

The highlight from the REST service set is the triple request method. This service enables building custom statements with specific subject, predicate or object properties, which are mapped into a SPARQL query to an instance's knowledge base (Figure 6-12). For example, http://bioinformatics.ua.pt/coeus/api/triple/coeus:hgnc_COL3A1/pred/obj/js returns a JSON object with all statements where the item **coeus:hgnc_COL3A1** (human gene COL3A1) is the subject from COEUS sample dataset. Similarly, http://bioinformatics.ua.pt/coeus/api/triple/sub/coeus:hasEntity/obj/xml returns XML detailing all subjects and objects related with a **coeus:hasEntity** predicate. In COEUS' ontology, this lists all concepts and respective entities.

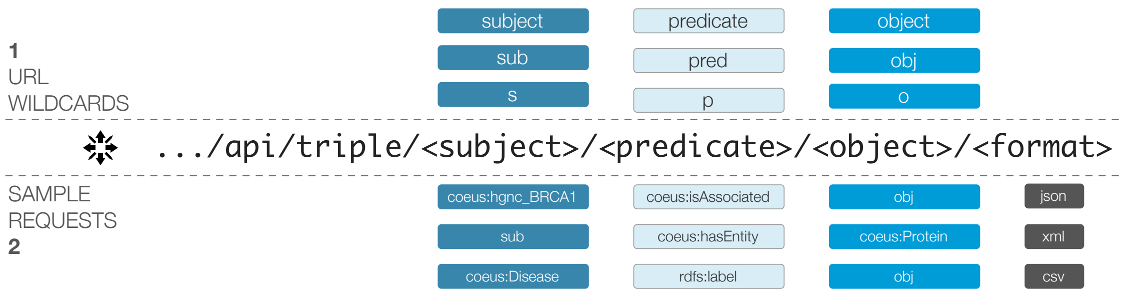


Figure 6-12. COEUS REST API summary. 1) Various wildcards can be combined to form valid requests and access all data in the knowledge base. 2) Sample REST requests highlighting the different output formats and wildcard use.

At last, the major advantage of using the available REST services is the access of data in multiple formats. Whereas requesting data in JSON format is optimal for lightweight web application development, one might need to import data in CSV format into an Excel sheet or transform XML content into a new database. This variety further increases COEUS' overall flexibility, improving its usage in modern application platform environments.

SPARQL

Another COEUS' API feature is the default SPARQL endpoint. With an open SPARQL endpoint, users or developers have full access to a seed's knowledge base, enabling complex queries and more insightful data retrieval operations.

Much like the set of REST services, the SPARQL endpoint also enables getting data in multiple formats, promoting its easier integration with client-side applications (discussed in the Advanced User Interactions section next). A form for querying each seed triplestore is available by default in all seeds at *../sparqler*. This form allows developers to test their SPARQL queries before including them in the application code.

LinkedData

Nowadays, the hottest topic in data sharing and interoperability is LinkedData. Through its multiple subdivisions, the LinkedData guidelines empower a completely interoperable knowledge ecosystem, where resources are directly accessible through their URIs and their semantic descriptions establish meaningful connections to other miscellaneous data types.

COEUS uses the **pubby** package to publish the knowledge base as LinkedData. A simple configuration file defines the connection properties to access the seed SPARQL endpoint and retrieve data. For each resource being browsed, the application issues a DESCRIBE to obtain all object relationships.

```
<!-- DESCRIBE <http://bioinformatics.ua.pt/coeus/resource/uniprot_P51587> -->
<?xml version="1.0"?>
```

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:dc="http://purl.org/dc/elements/1.1/"
    xmlns:owl2xml="http://www.w3.org/2006/12/owl2-xml#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns="http://bioinformatics.ua.pt/coeus/"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
       <isAssociatedTo
rdf:resource="http://bioinformatics.ua.pt/coeus/prosite_PS50138"/>
       <isAssociatedTo
rdf:resource="http://bioinformatics.ua.pt/coeus/mesh_D010051"/>
       <isAssociatedTo
rdf:resource="http://bioinformatics.ua.pt/coeus/interpro_IPR015525"/>
       <isAssociatedTo
rdf:resource="http://bioinformatics.ua.pt/coeus/pdb_3EU7"/>
       <dc:identifier>P51587</dc:identifier>
       <dc:source>P51587</dc:source>
       <isAssociatedTo
rdf:resource="http://bioinformatics.ua.pt/coeus/pdb_1N0W"/>
       <hasConcept
rdf:resource="http://bioinformatics.ua.pt/coeus/concept_UniProt"/>
       <rdfs:label>item_P51587</rdfs:label>
       <isAssociatedTo
rdf:resource="http://bioinformatics.ua.pt/coeus/hgnc_BRCA2"/>
       <isAssociatedTo
rdf:resource="http://bioinformatics.ua.pt/coeus/mesh_D001943"/>
       <dc:title>P51587</dc:title>
       <isAssociatedTo
rdf:resource="http://bioinformatics.ua.pt/coeus/mesh_D010190"/>
       <isAssociatedTo
rdf:resource="http://bioinformatics.ua.pt/coeus/mesh_D005910"/>
     </rdf:Description>
</rdf:RDF>
```

When these data are delivered through the web interface, users can explore LinkedData innate connections, which allow users to jump from object to object within the same seed, in an external seed or accessible through a normalized URI.

With the LinkedData interface, COEUS completes the interoperability features required to enhance modern service composition ecosystems. This ability to make the integrated and enriched data available in the Linked Open Cloud without complex configuration tasks or tricky deployment processes is a defining feature for COEUS, taking it further in semantic web for life sciences innovation.

Advanced User Interactions

Modern application development relies on advanced browser-based capabilities to deliver more compelling user interactions. The latest versions of all major browsers include powerful JavaScript processing engines, like Google's V8[54] or Mozilla's JagerMonkey[55], with

[54] http://code.google.com/p/v8/

outstanding performances on the client-side. This triggered an evolution on web-based application, making them able to deal with larger amounts of data and increasing their processing capabilities.

In addition, JavaScript development frameworks such as jQuery[56], MooTools[57] or SproutCore[58], include methods to further rival server-side data handling and computational capabilities. This allows for the development of increasingly interactive web applications, reducing the thin line that separates them from desktop-based applications.

It is important for COEUS to also take part in this emerging and fast-growing application trend. Therefore, COEUS includes a JavaScript library (available under *assets/js/sparqler.js*) that enables direct connections to each seed's SPARQL endpoint. With this, it is possible to ask queries to and process data directly from the knowledge base with a powerful querying language. Data are retrieved as a JSON object easily handled in JavaScript. This library further increases rapid application prototyping and interface development in COEUS.

6.2.4 Case Studies

Exploring Collected Data

A trivial case study can be setup to test the various elements composing COEUS APIs. For this matter, knowledge regarding the **breast cancer type 2 susceptibility protein** (UniProt accession number P51587) will be collected from COEUS sample dataset.

These results are obtained from the graph of relationships where a representation of this individual, mapped in COEUS' sample knowledge base as **coeus:uniprot_P51587**, is an active subject. The methods for accessing these data are detailed next.

→ **Java.** To obtain these data in Java, the ***getTriple()*** API method must be invoked, defining what filter to use and the desired XML output format.

```
/* Invoke getTriple("coeus:uniprot_P51587", "p", "o", "xml"); */
pt.ua.bioinformatics.API.getTriple(…);
```

→ **REST.** The desired protein data can be obtained, in CSV format for example, through a direct GET request to the public REST interface at http://bioinformatics.ua.pt/coeus/api/triple/coeus:uniprot_P51587/p/o/csv.

[55] https://wiki.mozilla.org/JaegerMonkey

[56] http://jquery.com/

[57] http://mootools.net/

[58] http://sproutcore.com/

- **SPARQL.** UniProt P51587 data can be queried from COEUS' SPARQL endpoint, available at http://bioinformatics.ua.pt/coeus/sparql, with the following query.

```
# SPARQL query to issue
PREFIX coeus: <http://bioinformatics.ua.pt/coeus/>
SELECT ?p ?o {coeus:uniprot_P51587 ?p ?o}
```

  Any query can be tested at http://bioinformatics.ua.pt/coeus/sparqler/.

- **LinkedData.** The requested protein data can be explored through a LinkedData browser pointed to http://bioinformatics.ua.pt/coeus/resource/uniprot_P51587. Additionally, the same address provides a summary view for regular web browsers.

Promoting a Federated Knowledge Ecosystem

The execution of federated SPARQL queries enables access to data across multiple sources in a single transaction. Whether data are locally stored or in a remote location, the query engine uses the **SERVICE** property to acknowledge where a specific question should be asked.

Every COEUS seed includes a SPARQL endpoint by default. With multiple seeds in place, it is fairly easy to perform queries across the various COEUS instances, inferring results on the fly. This virtual distributed knowledge network, the aforementioned knowledge garden, opens up immense data integration and interoperability possibilities. In modern national health information systems scenarios, launching multiple seeds with similar data models and targeted at regional subsets, originates a federated knowledge ecosystem. Applications can access each seed individually, cross data between two or more seeds, or have an holistic perspective over the entire knowledge garden.

A case study for COEUS' federation support regards the answers for following scientific question: *What are the PDB identifiers for the protein structures and the MeSH term identifiers associated with the human BRCA2 gene?*

To answer the proposed question, the federated query shown next links four distinct services, i.e. SPARQL endpoints. COEUS' default SPARQL is replicated three time to virtually simulate the query distribution. The query is processed in real time through the SPARQL endpoint, with the following steps:

1. The Diseasome SPARQL endpoint is queried to obtain the label for the human BRCA2 gene (**?label**).
2. The **?label** variable is passed to the first COEUS seed, acting as the selection clause for the gene and enabling access to a set of UniProt proteins associated with it (**?uniprot**).

3. The **?uniprot** variable is shared with the third and fourth SPARQL endpoints, where data regarding PDB identifiers (**?pdb**) and MeSH term identifiers (**?mesh**) is selected. This last request could be executed in a single query, but is divided to further demonstrate COEUS' federation capabilities.

```
   # Federated SPARQL query
   PREFIX dc: <http://purl.org/dc/elements/1.1/>
   PREFIX                    diseasome:                   <http://www4.wiwiss.fu-
berlin.de/diseasome/resource/diseasome/>
   PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
   PREFIX coeus: <http://bioinformatics.ua.pt/coeus/>

   SELECT ?pdb ?mesh
   WHERE{
       {
           SERVICE <http://www4.wiwiss.fu-berlin.de/diseasome/sparql>
           {
                   <http://www4.wiwiss.fu-
berlin.de/diseasome/resource/genes/BRCA2> rdfs:label ?label
           }
       }
       {
           SERVICE <http://bioinformatics.ua.pt/coeus/sparql>
           {
                   _:gene dc:title ?label .
                   _:gene coeus:isAssociatedTo ?uniprot
           }
       }
       {
           SERVICE <http://bioinformatics.ua.pt/coeus/sparql>
           {
                   ?uniprot coeus:isAssociatedTo ?pdb .
                   ?pdb coeus:hasConcept coeus:concept_PDB
           }
       }
       {
           SERVICE <http://bioinformatics.ua.pt/coeus/sparql>
           {
                   ?uniprot coeus:isAssociatedTo ?mesh .
                   ?mesh coeus:hasConcept coeus:concept_MeSH
           }
       }
   }
```

This, and any other federated queries, can be tested online at http://bioinformatics.ua.pt/coeus/sparqler/. Additionally, more complex queries can be built combining these data with any other SPARQL endpoint.

Despite being targeted at life sciences developers, COEUS can be used in various other real world settings. In either the corporate domain or TV networks, data are generated in large quantities and with complex innate relationships. While we do not envisage COEUS replacing already setup infrastructures in these areas, the framework is suitable for quickly deploying ad-hoc knowledge bases.

For example, a news channel web application can be built to aggregate information on a selected topic, from various media sources, in a single environment. With COEUS, content from Twitter, Facebook or any modern news site (using RSS/Atom feeds) can be quickly pulled into a new repository, enabling the creation of semantically richer applications for both web and mobile environments. Furthermore, the resulting dataset can be also used to improve existing applications. A new semantic layer can be incorporated in the client-side of modern web applications by querying and loading data in JSON format

Pharmaceutical companies can also use COEUS. A virtual scenario uses the COEUS platform with a well-designed ontology to create a Semantic Web-powered infrastructure to manage specific in-house datasets. Managing marketing results or large clinical trials data can be improved by establishing COEUS seeds, each with its own goals and needs, and allowing for future connections amongst these initially disparate data through COEUS' knowledge federation features.

6.2.5 Features and Usability

Rapid Application Development

COEUS' "Semantic Web in a box" approach streamlines the creation of new Semantic Web applications. The development of new semantic systems is highly associated with a steep learning curve and a myriad of technologies and tools to chose from. Although this variety is beneficial, it is also a characteristic of a still immature deployment environment. Unlike traditional relational database applications where the "technology path" is clearly outlined, with semantic web applications the adequate set of technologies and strategies continues to be chosen in *ad hoc* fashion.

COEUS provides the means for semantic web rapid application deployment by offering a single package comprising the set of tools required to develop a new application from scratch. Moreover, the application backend, the knowledge base, can be populated through advanced data integration wrappers that use flexible configuration ontology.

Incorporating the interoperability API with the integration features results in a framework that highly reduces the application "time-to-market". It is easy to get the data in the system. Likewise, it is easy to get the data out the system. This facilitates the creation of independent application platforms, supported by a comprehensive backend knowledge base that enables deploying to desktop, web or mobile systems. The API also permits coding client-side applications in any programming environment and using any framework, further improving COEUS flexibility and robustness.

Data Integration Platform

Data integration is the initial cornerstone for the COEUS framework. Its powerful resource integration capabilities enable the creation of customized niche-based data warehouses, powered by a semantic knowledge base. Distributed and heterogeneous data can be replicated or linked, taking advantage of semantic web's advanced data modelling capabilities to overcome schema mappings and internal wrappers for general data retrieval.

Data in CSV, SQL, XML or SPARQL formats are easily configured for integration, smoothing the transition from traditional data storage approaches to a modern semantic web reality. This migration is further improved through the advanced extract-transform-load warehousing features, providing a simple strategy for generating triple sets from any kind of data type. Moreover, custom plugins can be developed to match scenarios that do not fit COEUS' capabilities yet.

COEUS aids in the publishing of semantic web-powered knowledge bases, moving one step further to the envisaged view of the Internet as a semantically rich distributed knowledge network.

LinkedData & Semantic Services

Once data are integrated into a seed's knowledge base they are promptly available through various APIs. Firstly, the internal Java API layer hides away all complexities regarding semantic triple stores and data structures, offering a set of methods to retrieve data directly as an iterable result set.

Next, the REST services API encompasses simple GET-based methods to access data. The *triple* service offers a quick way to iteratively load all data into any application development environment.

The SPARQL endpoint is the most powerful interoperability feature. Besides supporting the LinkedData infrastructure and the client-side JavaScript library, it makes all data available through a standardized and efficient query engine. Complex queries can be asked and processed in command-line tools, web clients or desktop applications, further increasing the wide scope of COEUS' use. The SPARQL endpoint is also the underlying entry point for knowledge reasoning and inference features.

The LinkedData interface empowers the availability of integrated data in the most advanced data interoperability scenarios. With URLs uniquely and precisely identifying data descriptions numerous possibilities for service composition arise, taking the most out of connected, i.e. linked, data.

At last, the included JavaScript library enables creating best-of-breed user interactions, handling all data access and processing on the client-side. From the web application development perspective, this is the most interesting feature, as it enables the development of more responsive interactions in desktop- or mobile-based systems, taking web applications to the next usability level.

Knowledge Federation Framework

The challenge of federating knowledge scattered through multiple independent databases is also tackled in this framework. New seeds automatically launch SPARQL endpoints and LinkedData views, endowing developers with multiple ways to access and federate data.

Whether we are dealing with SPARQL-based federation or virtual LinkedData networks, data are inherently distributed and connected. With these technologies, anyone can launch his own customized and focused application ecosystem. In a COEUS' knowledge garden, the holistic view over all data empowers the sharing of knowledge amongst a scalable of numbers of peers, improving the federation of and facilitating access to data.

6.2.6 Future Developments

COEUS is an active project, published as open source with the purpose of captivating interest in new developments, thus creating a community surrounding the framework. Foreseen developments are focused on three main areas: improve the transition from monolithic systems to a semantic web environment, simplify the configuration of new seeds and provide new methods to input and output data from a seed.

Firstly, a **migration assistant** tool will be developed to smooth the transition from relational databases, CSV or XML structures into the semantic web paradigm. Leveraging on tools such as D2R we aim to create algorithms that read database structures and generate COEUS configuration models dynamically. For instance, automated processes to discover **Entity-Concept** organizations or internal data/object properties and import content on the fly will ease the creation of new seeds. Consequently, it will be much easier for bioinformaticians to transform their platforms and access all COEUS' integration and interoperability features.

The migration assistant will also feature **simplified configuration interfaces**. While now developers need to configure new seeds in Protégé or text-editor, a GUI-based setup & installation tool will be available in the future. This is aimed at non-expert bioinformatics developers that would rather fill in forms and click buttons than edit configuration settings *by hand*. Another step towards the simplification of seeds creation is the creation of a

COEUS virtual machine image pre-build with all required tools. In this case, the goal is to use a solution like TurnKey[59] to offer a disk image with the required application server, database and COEUS seed ready for deployment in a real-world scenario. Furthermore, this will also empower future COEUS integration with cloud-based developments.

Next, the API will also be augmented with **new applications and tools**. OS-specific applications and command-line tools for accessing COEUS' endpoints will be created. These will be an even better fit for a bioinformaticians research workflow. For instance, old-school biologists traditionally use basic shell scripts to perform data filtering and enriching operations. These can be enhanced with access to a COEUS knowledge base allowing the integration of state-of-the-art integrative datasets with legacy tools. Another opportunity concerns COEUS service composition. REST services will be improved and new ones developed to ease the process of combining COEUS services in Taverna workflows. With Taverna as the *de facto* workflow platform, it is advisable to foster COEUS use in this service composition environment.

6.3 Discussion

6.3.1 Ad-hoc Software Solutions VS Rapid Application Development

Developing tailored *ad hoc* solutions is the current practice in bioinformatics. Solutions like the ones highlighted in chapters 4 and 5 (EU-ADR Web Platform and WAVe) play a fundamental role in the evolution of the way bioinformatics software is developed. In spite of the recent turn of events in the innovative technologies side, where previously built packages are preferred over deployment from scratch, we must realize that bioinformaticians are not "regular" developers. The traditional bioinformatician's background usually lacks computer science skills, such as database management, modelling or object-oriented programming. Since most stakeholders fit this profile, it is easy to understand the biased focus on building new systems from scratch, paying little to no consideration to existing platforms, frameworks or programming libraries.

On the other end of the spectrum is the use of rapid application development strategies. By considering RAD ideals in a very broad sense, we observe that its practices are already being used in the majority of innovative technological platforms. Reusable assets are being used more often whether in the form of fully-fledged application frameworks, user

[59] http://www.turnkeylinux.org/

interface bootstrapping packages or simple external libraries. Over time, the inclusion of these components in new systems became easier, empowering the creation of new tools and disseminating the adoption of RAD ideals. The created strategies that empower COEUS build on this growing use of RAD principles, aiming at its use to create innovative biomedical applications.

6.3.2 Enhancing Rapid Application Development

The overall concept of RAD strategies for bioinformatics is still in its infancy. Molgenis is one successful case in the area, with a robust framework for launching new bioinformatics applications very quickly. The room for improvements over general RAD and bioinformatics-specific RAD is tied to two domains: integration & interoperability research and the semantic web paradigm.

RAD frameworks are not prone to facilitating the integration of data from external resources. Whereas the ability to quickly deploy data stores is omnipresent, RAD frameworks lack the features required to easily populate those data stores. The multiple challenges associated with the integration of data in any field, detailed along this thesis, are cumbersome for bioinformatics developers. Hence, the inclusion of integration features is deemed vital for bioinformatics RAD frameworks. Collecting and transforming data from CSV, SQL or XML files into a centralized knowledge base is a must-have feature in a field riddled with data heterogeneity and distribution. COEUS achieves this through a flexible integration engine, allowing the mapping of existing content into any ontology, and storing generated triples in a centralized knowledge base. Continuing DynamicFlow and WAVe's pursuit of the best service description strategy for data integration, COEUS uses an adaptive ontology to organize and configure a set of integrated resources.

As previously mentioned, the semantic web paradigm adoption and acceptance by the life sciences community is growing and it emerges as a viable alternative to lead biomedical software to a new level with tighter integration and better interoperability. The applicability of semantic web's ideals fits perfectly the complex life sciences challenges set. However, the steep learning curve associated with semantic web technologies is drawing users away from this new world. As such, the opportunity arises for the inclusion of semantic web technologies and features within a rapid application development package. For this matter, bioinformaticians must think about triplestores instead of relational databases, about SPARQL endpoints instead of SQL hosts or about LinkedData instead of SOAP-based data exchanges. The semantic web empowers a new services layer that allows the creation of truly federated intelligent data networks. Combining LinkedData with SPARQL endpoints we can connect and exploit the wealth of data from miscellaneous data stores. As stated in the initial requirements, COEUS includes, by default, a SPARQL endpoint and support for LinkedData views, enabling truly semantic access to data collected in a single seed or federated from multiple COEUS instances.

6.3.3 A Framework for Semantic Bioinformatics Software

The next-generation of bioinformatics software will be empowered by the combination of two grand innovations that are diluting the boundaries between computer and life sciences.

On the computer science standpoint, the adoption of agile strategies to develop new applications is pushing forward the adoption of generic rapid application development methods, from reusable programming packages to user interface prototype building. For life sciences, enhanced biomedical semantics are the cornerstone for a better understanding of our human condition. While it will not solve all problems in bioinformatics, the semantic web emerges as the most viable alternative to build the next-generation of biomedical knowledge.

The COEUS framework is our approach to tackle these challenges and produce a next-generation semantic web rapid application development framework. The innovative "Semantic Web in a Box" approach encloses four major pioneering roles.

- → The adoption of **rapid application development** strategies in COEUS endows developers with the tools to quickly build new application ecosystems targeting any deployment environment.
- → COEUS is a **semantic data integration platform** enabling the acquisition and translation of heterogeneous data from distributed resources intro a centralized knowledge base.
- → COEUS provides **Semantic Web and LinkedData services** by default. This ensures the interoperability of integrated data with any external system through open standard methods [13]. Moreover, with a semantic knowledge base in place, support for reasoning and inference strategies is facilitated.
- → COEUS enables the **federation** of gathered knowledge through comprehensive APIs [12]. The SPARQL endpoint and LinkedData interfaces empower querying and reasoning over multiple COEUS instances.

The COEUS framework is an open-source project. Documentation and code samples are available online at http://bioinformatics.ua.pt/coeus/ [14].

7. A COEUS INSTANCE

"But my intellectual development was retarded, as a result of which I began to wonder about space and time only when I had already grown up."
- Albert Einstein

Many bioinformatics platforms are emerging within the constantly evolving life sciences field that satisfy most integration and interoperability requirements. The main advantage of this evolution is that developers do not need to rebuild entire knowledge systems and data infrastructures from scratch. It is possible to reuse and recombine existing components to form entirely new software systems as an answer to the latest challenges. Furthermore, with the COEUS framework in place, we have the tools required to launch a new semantic web application with minimal effort.

Continuing our research within the individualized healthcare field, we tackle the study of rare diseases with the development of a new version for the Diseasecard platform. The personal health implications behind rare diseases are seldom considered in widespread medical care. The low incidence rate and complex treatment process makes rare disease research an underrated field in the life sciences. Diseasecard, an online portal containing thousands of pointers to rare disease resources, was developed to aid rare disease investigators. However, the uncontrollable evolution of data and services in the field, united with an aging legacy code, triggers the need for a new release.

Not only was Diseasecard's server-side code dated, the user interface was also in the need of a facelift to better suit the current generation of web applications. Hence, Diseasecard appeared as the perfect benchmark for COEUS' developments. An initial prototype of the new Diseasecard, COEUS first public instance, is available at http://bioinformatics.ua.pt/dc4/.

In this chapter we introduce the new Diseasecard platform and discuss the details behind its development. Starting with a brief analysis of the legacy Diseasecard portal, we cover the easy process of creating a new COEUS seed, from the construction of a rare disease knowledge base to the details of Diseasecard's new user interface.

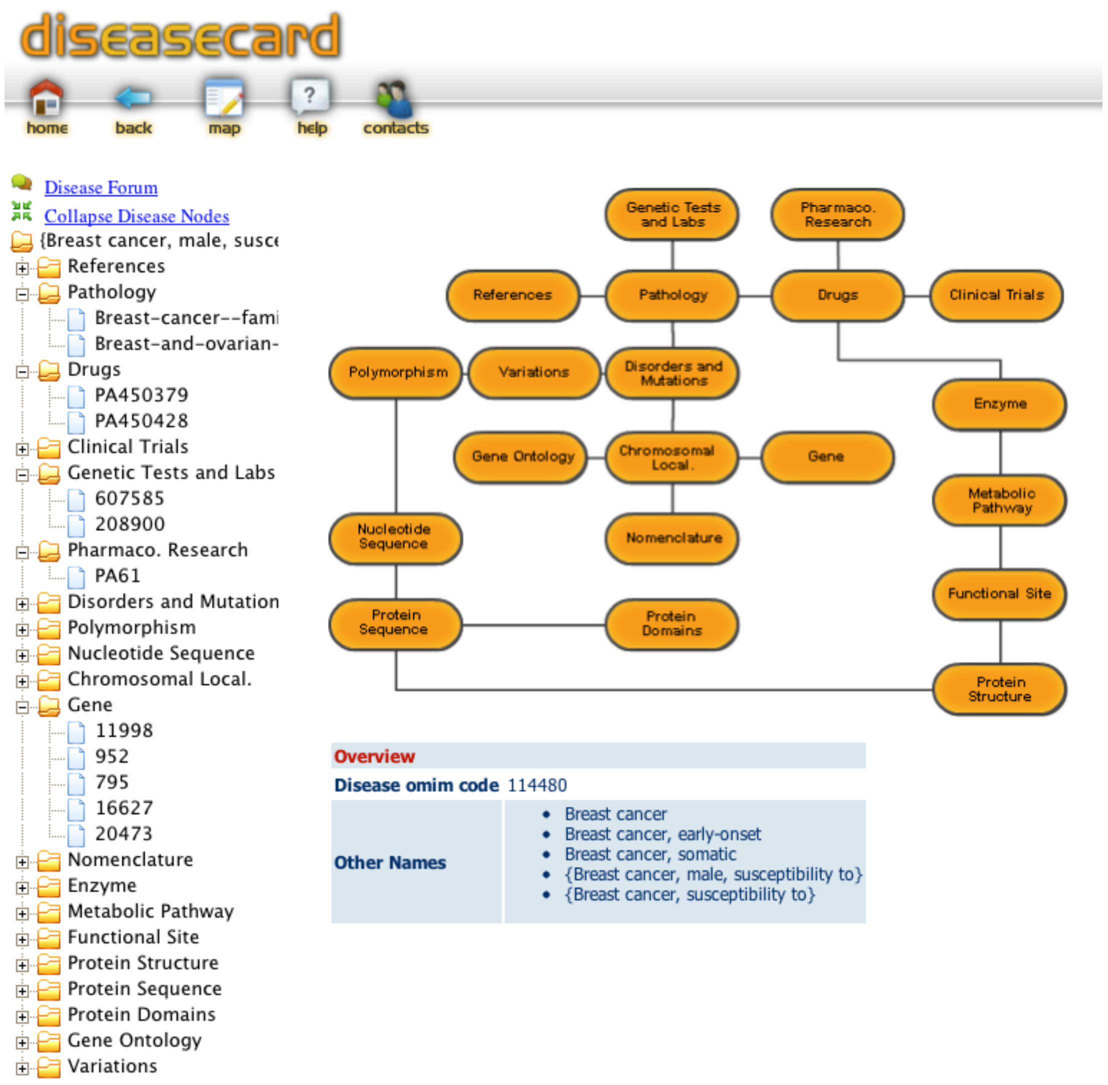
7.1 Improving Rare Diseases Research

Rare diseases' particular conditions hold the strongest relations between genotypes and phenotypes. Understanding gene-disease associations is a fundamental goal for bioinformatics research, especially at rare disease level, where the genotype-phenotype connections are limited to a small set of genes. Rare diseases are particular conditions that affect at most 1 in 2000 patients [215]. The European Organization for Rare Diseases (EURORDIS) estimates that there are approximately 6000 to 8000 rare diseases, affecting about 6% to 8% of the population. Within these, about 80% are caused by genetic disorders. Due to the reduced incidence of each individual disease, it is difficult for patients to find support, both at clinical and psychological levels [216]. Some of these chronic diseases hinder the patients' quality of life and cause serious damage or disability in social terms. The existence of a small number of patients for each rare disease also delays the creation of adequate research studies, as it is difficult to identify and coordinate a relevant cohort [168, 217]. Despite the low statistic impact regarding these diseases, the combined amount of patients suffering from one of these rare diseases is considerably high.

7.1.1 Diseasecard's Legacy

Diseasecard was developed to improve rare disease research and education. It is a web portal that uses link integration strategies to establish connections to a myriad of external resources. The goal is to provide a central workspace where users can explore available connections to assess rare diseases underlying genotype, associated proteins or pathways, known drugs, ongoing clinical trials or relevant literature (Figure 7-1).

Initial Diseasecard developments date back to 2004. At that stage, the data acquisition strategy relied on web crawling to discover links for the various data types integrated in the database, and static HTML pages contained most of the Internet content. The idea of providing services to access data was not mature enough yet and most data was still published in CSV files or similar text-based tabular formats. Diseasecard's platform uses a link integration engine, pre-configured with a navigation map that teaches the system what links to collect and what links to follow for further crawling. Whilst this strategy worked for the 2004 timeline, it is currently totally inadequate.



Disease Forum
Collapse Disease Nodes
{Breast cancer, male, susc
- References
- Pathology
  - Breast-cancer--fami
  - Breast-and-ovarian-
- Drugs
  - PA450379
  - PA450428
- Clinical Trials
- Genetic Tests and Labs
  - 607585
  - 208900
- Pharmaco. Research
  - PA61
- Disorders and Mutation
- Polymorphism
- Nucleotide Sequence
- Chromosomal Local.
- Gene
  - 11998
  - 952
  - 795
  - 16627
  - 20473
- Nomenclature
- Enzyme
- Metabolic Pathway
- Functional Site
- Protein Structure
- Protein Sequence
- Protein Domains
- Gene Ontology
- Variations

Overview

Disease omim code	114480
Other Names	• Breast cancer • Breast cancer, early-onset • Breast cancer, somatic • {Breast cancer, male, susceptibility to} • {Breast cancer, susceptibility to}

Figure 7-1. Diseasecard workspace for Alzheimer's Disease, OMIM code 104300 (from legacy Diseasecard[60]).

7.1.2 Collecting Rare Disease Information

Much like the human variome scenario, rare disease information is scattered through multiple non-exchangeable data sources. In a sense, Diseasecard development is a common "by the books" service composition problem. With heterogeneous rare disease data fragmented through multiple independent resources, new strategies must be devised to collect it and make it available for other tools.

In a world with personalized medicine and individual healthcare as primary research topics, advanced integration and interoperability tools are essential. The huge amounts of data are meaningless unless they are interconnected with rich relationships. Moreover, data integration in bioinformatics has been mostly focused on genotype data. Nowadays,

[60] http://bioinformatics.ua.pt/diseasecard/evaluateCard.do?diseaseid=104300

the goal is to balance the scale. We need to access the increasing quantity of clinical phenotype data and combine it with existing rich genotype resources, empowering a new knowledge reasoning level.

Despite dated, Diseasecard's initial approach already preconceived this much-needed integration from genotypes to phenotypes. However, with the appearance of WAVe we already provide an alternative geared towards genotype data. Hence, the new Diseasecard is much more directed towards phenotype information. This demands establishing a new rare disease relationship network that in spite of being based on the original Diseasecard, further specializes it with another filtering layer.

7.2 The New Diseasecard

Developing a new Diseasecard version was an entirely different task from developing a new application from scratch. The complex requirements analysis or data modelling tasks were already executed and documented for the original portal. Likewise, mock-ups were not required for the interface design as the idea was to improve on existing interactions and to make the lower number of changes as possible, always without disrupting the tree-based and map-based navigation metaphors.

Supported by previous requirements' comprehension, we only needed to update Diseasecard's internal data model to fit the COEUS seed configuration. This requires organizing data in the **Entity-Concept-Item** tree structure and defining the integration properties. Whereas the original Diseasecard used a map-based navigation model to crawl for identifiers, parsing HTML content from webpages dynamically, the new COEUS-based integration engine uses web services and databases to load data and generate a similar, yet richer, rare disease knowledge network.

In addition, this rare disease knowledge network is available for reasoning and inference. The new data relationships allow denser knowledge connections, further enabling the success and availability of reasoning features, which may result in deeper rare diseases insights. Through these new connections we can also infer new knowledge. As such, by semantically integrating data from miscellaneous heterogeneous resources, we are empowering the discovery of new direct links from genotypes to phenotypes in rare diseases.

Regarding the web application, we tried to maintain the user interactions already present in the legacy Diseasecard. Using the same metaphor, the new Diseasecard delivers an improved user experience. The navigation tree is smoother and more complete and leaf links trigger the **Live View** feature. The latter promotes accreditation and ownership of original work, loading the external resource directly within Diseasecard's workspace, just like in WAVe.

At last, the legacy Diseasecard included a navigation map. In this map, users could identify which data types were available for each specific disease. In the new Diseasecard, this key static interaction component was replaced by a dynamic identifier map. Besides being a more interactive tool, the new navigation map enables accessing the external resources directly, without additional navigation tree mouse clicks.

7.2.1 Application Setup

Data Model

Following COEUS "reuse instead of rewrite" motto, the new Diseasecard's data model reuses existing schemas internally. Using COEUS seed configuration and taking advantage of existing ontologies and models for internal usage is enough to organize collected data.

For each individual item, such as a UniProt protein or an OMIM disease, we only need to store its identifier. Hence, we can reuse the *identifier* term from the Dublin Core ontology [218]. In Diseasecard, each Item individual has a **dc:identifier** data property, matching a string with the external identifier. COEUS enables reusing any kind of property, liberating our knowledge base from strict data models. Another example is the **rdfs:label** property, obtained from the RDF schema ontology that is used to label each individual object in Diseasecard, whether it is an Entity, an Item or a Resource.

This "reuse instead of rewrite" fits most required properties. Nevertheless, to further enhance user interactions new relationships were required. Diseases may have multiple names and OMIM's internal structure makes distinctions from phenotype and genotype identifiers. To this end, new data and object predicates were created, as listed in Table 7-1. With the set of integrated resources in place and the new model designed, Diseasecard was ready to be launched as a new COEUS seed.

Table 7-1. List of new predicates in Diseasecard ontology.

PREDICATE	RELATIONSHIP	DESCRIPTION
	Object Properties	
hasGenotype	Disease to Disease	Connects a Disease phenotype entry with its associated genotypes.
hasPhenotype	Disease to Disease	Connects a Disease genotype entry with its associated phenotypes.
	Data Properties	
chromossomalLocation	to String	Chromossomal location information (read from MorbidMap).
genotype	to Boolean	True if Disease is a genotype.
name	to String	Disease name.
omim	to String	Disease OMIM accession number.
phenotype	to Boolean	True if Disease is a phenotype.

A New COEUS Seed

As mentioned, Diseasecard is the first COEUS seed. To launch a new COEUS the initial step is to download or clone COEUS' source code into a new development workspace. Java, an Apache Tomcat server and a MySQL database must be in place to set up the new system. As mentioned in the previous chapter, the configuration involves three files:

→ The Diseasecard model is transposed to a new ontology[61], including the new data and object properties. This file can be created and managed using Protégé.

→ ***Config.js***, the seed configuration file, contains the details for the new Diseasecard application properties. These basic properties define where further configurations, such as the seed ontology, the setup files or MySQL database connections are stored.

```
# Diseasecard application properties file
{
    "config": {
        "name": "Diseasecard",
        "description": "Diseasecard v4",
        "keyprefix":"coeus",
        "version": "4.0",
        "ontology": "http://bioinformatics.ua.pt/dc4/diseasecard.owl",
        "setup": "dc4_setup.rdf",
        "sdb":"dc4_sdb.ttl",
        "predicates":"dc4_predicates.csv",
        "built": true,
        "debug": false,
        "environment": "testing"
    }, … }
```

[61] http://bioinformatics.ua.pt/dc4/diseasecard.owl

- → The seed setup file, ***dc4_setup.rdf***, includes the internal data structure and resource configurations, defining how to connect to and exploit resources in Diseasecard's network.

Additionally, three files must be updated with knowledge base connection properties: one for **Jena**, a second for **Joseki** and a third for **pubby**. **Jena** and **Joseki** definitions are similar and include the seed's MySQL database connection properties. The third file, for **pubby**, includes the LinkedData configurations such as the system SPARQL endpoint and internal ontology base URIs.

Resource Configuration

COEUS allows collecting data in miscellaneous formats from local or remote data sources. For Diseasecard's seed, we are looking to build a semantically powerful data network. Hence, we need to obtain a huge amount of identifier mappings. These mappings are usually available as CSV or XML files in some sort of FTP file server.

Figure 7-2 shows Diseasecard's data integration graph, detailing how COEUS integration engine moves from one resource to the next. The starting resource uses a custom connector plugin, processing OMIM's morbid and gene maps.

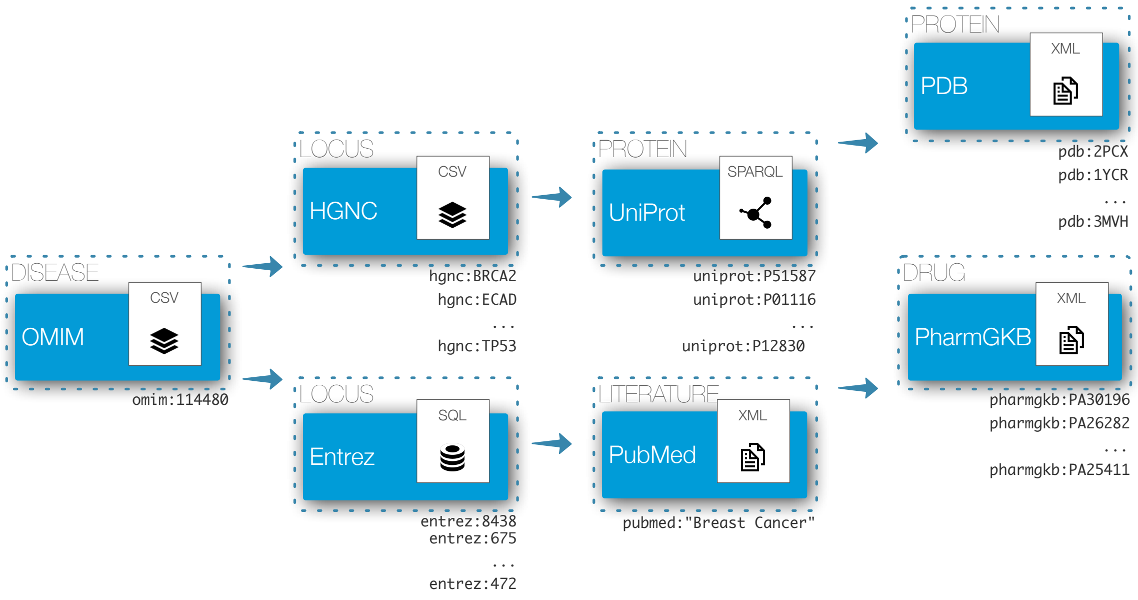


Figure 7-2. Subset of Diseasecard's integration graph. This seed uses COEUS's flexible integration engine to acquire data from heterogeneous and distributed CSV, XML, SQL and SPARQL resources. The integration process generates a rich data network. For example, starting with Breast cancer in OMIM (114480) we obtain multiple genes from HGNC database (BRCA2, TP53...), which are used individually next to obtain a list of UniProt identifiers (P51587, P12830...). From UniProt data we also extract PharmGKB (PA30196, PA26282...) and PDB (2PCX, 1YCR...) identifiers, among others. This process continues until data are fully integrated for all resources in Diseasecard's configuration.

Each resource is configured individually in the local setup file. Once again, Protégé use is advised to build this file, making it fairly easy to edit COEUS setup. In this case, each concept corresponds to an external resource, being it a database or application. The following simplified code snippets highlight the *Protein* **Entity**, the *UniProt* **Concept** and its respective **Resource**.

```
<!-- Protein Entity configuration -->
<owl:NamedIndividual rdf:about="http://bioinformatics.ua.pt/coeus/entity_Protein">
   <rdf:type rdf:resource="http://bioinformatics.ua.pt/coeus/Entity"/>
   <rdfs:label rdf:datatype="&xsd;string">entity_protein</rdfs:label>
   <dc:title rdf:datatype="&xsd;string">Protein</dc:title>
   <isEntityOf
rdf:resource="http://bioinformatics.ua.pt/coeus/concept_InterPro"/>
   <isEntityOf rdf:resource="http://bioinformatics.ua.pt/coeus/concept_PDB"/>
   <isEntityOf rdf:resource="http://bioinformatics.ua.pt/coeus/concept_PROSITE"/>
   <isEntityOf rdf:resource="http://bioinformatics.ua.pt/coeus/concept_UniProt"/>
   <isIncludedIn
rdf:resource="http://bioinformatics.ua.pt/coeus/seed_Diseasecard4"/>
</owl:NamedIndividual>

<!-- UniProt Concept configuration -->
<owl:NamedIndividual
rdf:about="http://bioinformatics.ua.pt/coeus/concept_UniProt">
   <rdf:type rdf:resource="http://bioinformatics.ua.pt/coeus/Concept"/>
   <rdfs:label rdf:datatype="&xsd;string">concept_uniprot</rdfs:label>
   <dc:title rdf:datatype="&xsd;string">UniProt</dc:title>
   <hasEntity rdf:resource="http://bioinformatics.ua.pt/coeus/entity_Protein"/>
   <isExtendedBy
rdf:resource="http://bioinformatics.ua.pt/coeus/resource_DrugBank"/>
   <isExtendedBy
rdf:resource="http://bioinformatics.ua.pt/coeus/resource_InterPro"/>
   <isExtendedBy rdf:resource="http://bioinformatics.ua.pt/coeus/resource_MeSH"/>
   <isExtendedBy rdf:resource="http://bioinformatics.ua.pt/coeus/resource_PDB"/>
   <isExtendedBy
rdf:resource="http://bioinformatics.ua.pt/coeus/resource_PROSITE"/>
   <hasResource
rdf:resource="http://bioinformatics.ua.pt/coeus/resource_UniProt"/>
</owl:NamedIndividual>

<!-- UniProt Resource configuration -->
<owl:NamedIndividual
rdf:about="http://bioinformatics.ua.pt/coeus/resource_UniProt">
   <rdf:type rdf:resource="http://bioinformatics.ua.pt/coeus/Resource"/>
   <rdfs:label>resource_uniprot</rdfs:label>
   <order rdf:datatype="&xsd;integer">2</order>
   <dc:title rdf:datatype="&xsd;string">UniProt</dc:title>
   <method rdf:datatype="&xsd;string">cache</method>
   <dc:publisher rdf:datatype="&xsd;string">csv</dc:publisher>
   <endpoint              rdf:datatype="&xsd;string">http://www.genenames.org/cgi-
bin/hgnc_downloads.cgi?title=HGNC+output+data&amp;hgnc_dbtag=on&amp;col=md_prot_id
&amp;status=Approved&amp;status=Entry+Withdrawn&amp;status_opt=2&amp;level=pri&amp
;where=gd_app_sym+LIKE+%27#replace#%27&amp;order_by=gd_app_sym_sort&amp;limit=&amp
;format=text&amp;submit=submit&amp;.cgifields=&amp;.cgifields=level&amp;.cgifields
=chr&amp;.cgifields=status&amp;.cgifields=hgnc_dbtag</endpoint>
    <extends rdf:resource="http://bioinformatics.ua.pt/coeus/concept_HGNC"/>
    <isResourceOf
rdf:resource="http://bioinformatics.ua.pt/coeus/concept_UniProt"/>
    <hasKey rdf:resource="http://bioinformatics.ua.pt/coeus/csv_UniProt_id"/>
    <loadsFrom rdf:resource="http://bioinformatics.ua.pt/coeus/csv_UniProt_id"/>
</owl:NamedIndividual>
```

Once all resources are configured correctly, the actual integration process starts, populating Diseasecard's knowledge base. This is envisaged as a spiralled iterative process, where each iteration fine-tunes the previous one.

From OMIM's MorbidMap to 5 Million Triples

Diseasecard adopts a targeted warehousing strategy. This means that data are integrated once and stays static until the following build process. Accordingly, the data import and translation process gathers all data from external resources in a single centralized Diseasecard knowledge base. In Diseasecard, this process starts with a custom *connector* plugin to process OMIM's data, traversing the dependency graph for all configured resources iteratively.

During this data import process triples are generated from external data. Adding a new semantic layer on top of existing data results in an augmented dataset. COEUS adds several metadata relationships to each item along with the configured resource properties. Moreover, connections are established from items to concepts, from concepts to items and amongst items. These rich relationships are what make semantic knowledge bases so powerful. Whereas in a CSV file we have a set of columns with text, with the move to a semantic environment all data are interconnected, generating a richer dataset. The same is true for SQL databases, where foreign key relationships or table/column names are mapped to new properties, resulting in more metadata, more relationships and more triples.

OMIM's morbid map has around 5600 entries related to a gene map with about 12800 entries. From these maps, the graph proceeds to link multiple entities and concepts, increasing the amount of data exponentially. The current Diseasecard build accounts for almost 5 million triples. Leveraging on the big data network and the additional metadata, this number grows constantly as each new resource is integrated. Despite the 5 million triples, the knowledge base only stores around 1.5 million distinct individuals. This is further proof that collected data are deeply intertwined, resulting in a very dense graph.

Building Diseasecard's knowledge base highlighted some issues with COEUS building process. The performance is severely hindered by the repetitive connections to external web services or by complex SQL queries. Executing the build process as a single task in a single thread takes to long to be acceptable. This fostered the development of a basic multithreaded integration solution. The multithreaded strategy involves processing the various resources at different levels based on the configured dependency graph - Figure 7-3. Diseasecard's multithread integration solution was promptly integrated within COEUS. Nevertheless, foreseen developments will focus on improving the code implementation for this feature.

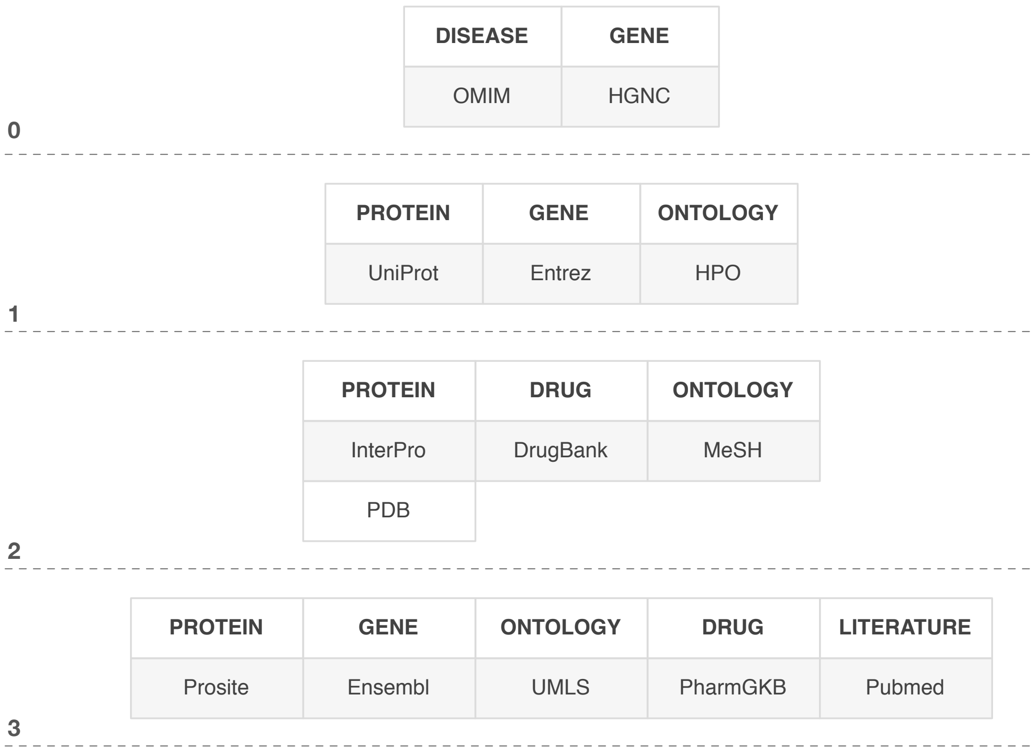
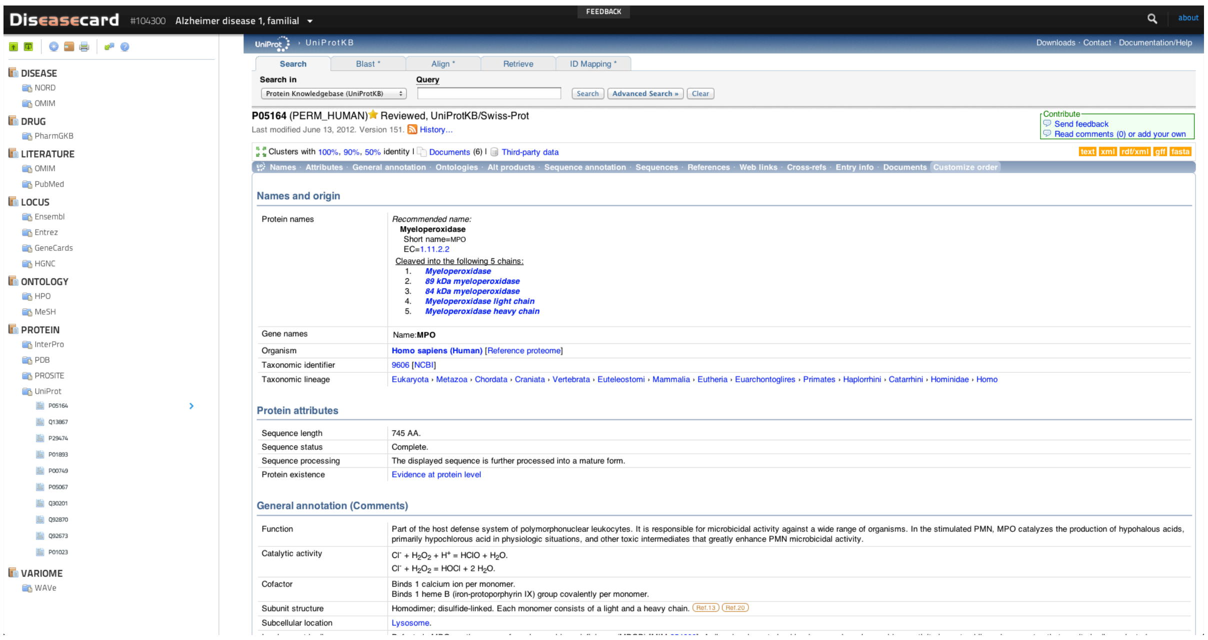


Figure 7-3. Diseasecard build process levels. The multiple stages use a multithreaded approach to load data from external resources, significantly improving the process efficiency and performance.

Building the New Diseasecard's User Interface

With Diseasecard's triplestore populated, interoperability services are enabled. This means that the default API methods (Java, SPARQL, REST, LinkedData, JavaScript) are ready for us. Miscellaneous services were created to access and retrieve data required by the client-side application (Figure 7-4).

Modern web applications employ new user interaction approaches that require flexible server-side code and intelligent client-side code. On the server-side, the application controller must offer easy access points to all data and, if possible, in custom formats ready for use in the web application. The client-side should handle most of the payload for processing data. This does not mean that browsers will perform intensive data processing or transformation activities, but should rely more on asynchronous data exchanges.



P05164 (PERM_HUMAN) Reviewed, UniProtKB/Swiss-Prot
Last modified June 13, 2012. Version 151. History...

Clusters with 100%, 90%, 50% identity | Documents (6) | Third-party data

Names and origin

Protein names	*Recommended name:* **Myeloperoxidase** Short name=MPO EC=1.11.2.2 Cleaved into the following 5 chains: 1. ***Myeloperoxidase*** 2. ***89 kDa myeloperoxidase*** 3. ***84 kDa myeloperoxidase*** 4. ***Myeloperoxidase light chain*** 5. ***Myeloperoxidase heavy chain***
Gene names	Name:**MPO**
Organism	**Homo sapiens (Human)** [Reference proteome]
Taxonomic identifier	9606 [NCBI]
Taxonomic lineage	Eukaryota › Metazoa › Chordata › Craniata › Vertebrata › Euteleostomi › Mammalia › Eutheria › Euarchontoglires › Primates › Haplorrhini › Catarrhini › Hominidae › Homo

Protein attributes

Sequence length	745 AA.
Sequence status	Complete.
Sequence processing	The displayed sequence is further processed into a mature form.
Protein existence	Evidence at protein level

General annotation (Comments)

Function	Part of the host defense system of polymorphonuclear leukocytes. It is responsible for microbicidal activity against a wide range of organisms. In the stimulated PMN, MPO catalyzes the production of hypohalous acids, primarily hypochlorous acid in physiologic situations, and other toxic intermediates that greatly enhance PMN microbicidal activity.
Catalytic activity	$Cl^- + H_2O_2 + H^+ = HClO + H_2O$. $Cl^- + H_2O_2 = HOCl + 2\ H_2O$.
Cofactor	Binds 1 calcium ion per monomer. Binds 1 heme B (iron-protoporphyrin IX) group covalently per monomer.
Subunit structure	Homodimer; disulfide-linked. Each monomer consists of a light and a heavy chain. Ref.13 Ref.20
Subcellular location	Lysosome.

Figure 7-4. The new Diseasecard workspace for Alzheimer's Disease, OMIM code 104300[62].

Diseasecard implementation uses COEUS API to apply these modern data access paradigms. Relying on internal Java methods, Diseasecard includes multiple actions to mediate data access from the client application to the seed's knowledge base. For instance, a method for retrieving a data network associated with a single OMIM code (104300, Alzheimer's Disease) is available at http://bioinformatics.ua.pt/dc4/content/104300.js. This returns a JSON object with the disease information that is used to generate, on the browser, both the sidebar navigation tree and central navigation map. Similarly, data requests to the REST *triple* service are used to load disease synonyms (http://bioinformatics.ua.pt/dc4/api/triple/coeus:omim_104300/dc:description/obj/js).

These interactions use COEUS API and enable a faster web application, with smaller data requests and improved responsiveness. Comparing the sidebar navigation tree in the legacy Diseasecard with the new one, users had to wait for the entire page to be processed on the server and then sent to the browser before the webpage actually appears. In the new version, the page loads completely and provides adequate feedback to users while the navigation tree and map are being loaded.

Another welcome addition is the inclusion of an easier bookmarking tool. In the current version, when **Live View** is triggered, the page URL is updated in the browser, enabling the creation of bookmarks pointing directly to an external resource within a disease context in Diseasecard (similar to WAVe's **UniversalAccess**).

[62] http://bioinformatics.ua.pt/dc4/disease/104300

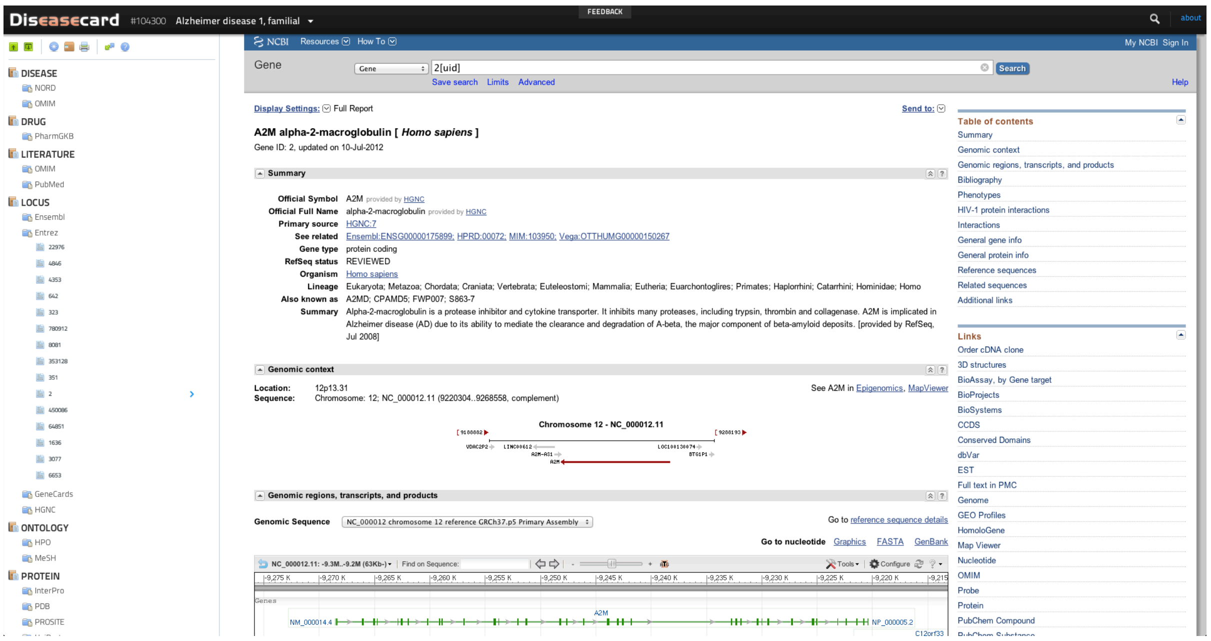
With a re-engineered server side it was also essential to revamp Diseasecard's interface. With a sleeker design, the new Diseasecard has improved usability and delivers a more fulfilling experience to end-users.

7.2.2 Features and Usability

Context-based Navigation

Diseasecard is a unique alternative for browsing biomedical rare disease information in a centralized environment. The context-based navigation approach enables exploring a variety of resources associated to a single disease and also browsing disease synonyms. With these two complimentary perspectives all significantly relevant resources associated to one or more diseases are a couple clicks away.

The workspace includes two disease data network navigation alternatives. The left sidebar includes a tree to quickly access all links with a familiar metaphor. The central area displays a circular navigation map, pointing to all individual identifiers. This map is an outstanding improvement from what was previously available, making it one of Diseasecard's key features. Both the navigation tree and map trigger the **Live View** feature (Figure 7-5).



A2M alpha-2-macroglobulin [*Homo sapiens*]
Gene ID: 2, updated on 10-Jul-2012

Summary

Official Symbol	A2M provided by HGNC
Official Full Name	alpha-2-macroglobulin provided by HGNC
Primary source	HGNC:7
See related	Ensembl:ENSG00000175899; HPRD:00072; MIM:103950; Vega:OTTHUMG00000150267
Gene type	protein coding
RefSeq status	REVIEWED
Organism	Homo sapiens
Lineage	Eukaryota; Metazoa; Chordata; Craniata; Vertebrata; Euteleostomi; Mammalia; Eutheria; Euarchontoglires; Primates; Haplorrhini; Catarrhini; Hominidae; Homo
Also known as	A2MD; CPAMD5; FWP007; S863-7
Summary	Alpha-2-macroglobulin is a protease inhibitor and cytokine transporter. It inhibits many proteases, including trypsin, thrombin and collagenase. A2M is implicated in Alzheimer disease (AD) due to its ability to mediate the clearance and degradation of A-beta, the major component of beta-amyloid deposits. [provided by RefSeq, Jul 2008]

Genomic context

Location: 12p13.31
Sequence: Chromosome: 12; NC_000012.11 (9220304..9268558, complement)

See A2M in Epigenomics, MapViewer

Genomic regions, transcripts, and products

Genomic Sequence NC_000012 chromosome 12 reference GRCh37.p5 Primary Assembly

Go to reference sequence details

Go to nucleotide Graphics FASTA GenBank

Figure 7-5. Diseasecard's workspace for Alzheimer's Disease, OMIM code 104300, highlighting Entrez Gene entry A2M for *Homo sapiens*[63] in LiveView.

[63] http://bioinformatics.ua.pt/dc4/disease/104300#entrez:2

This approach was used initially in the legacy Diseasecard and was enhanced for WAVe. The newest Diseasecard version further improves WAVe's approach, making **Live View** more interactive, responsive and usable.

Resources Relationship Graph

To correctly explore Diseasecard's huge amount of data and relationships we could not rely on a static navigation system or a non-scalable navigation tree. This rich rare disease resource relationship graph provides a unique wealth of direct and indirect connections. Hence, a suitable approach for displaying these relationships was required. Our choice set on the JavaScript InfoVis Toolkit[64] framework (JIT). This framework combines the power of client-side data handling with a collection of visualization approaches based on JavaScript JSON objects and manipulations on the DOM canvas. Figure 7-6 shows Diseasecard using JIT to expose a disease map to users in a simple aesthetically pleasing way.

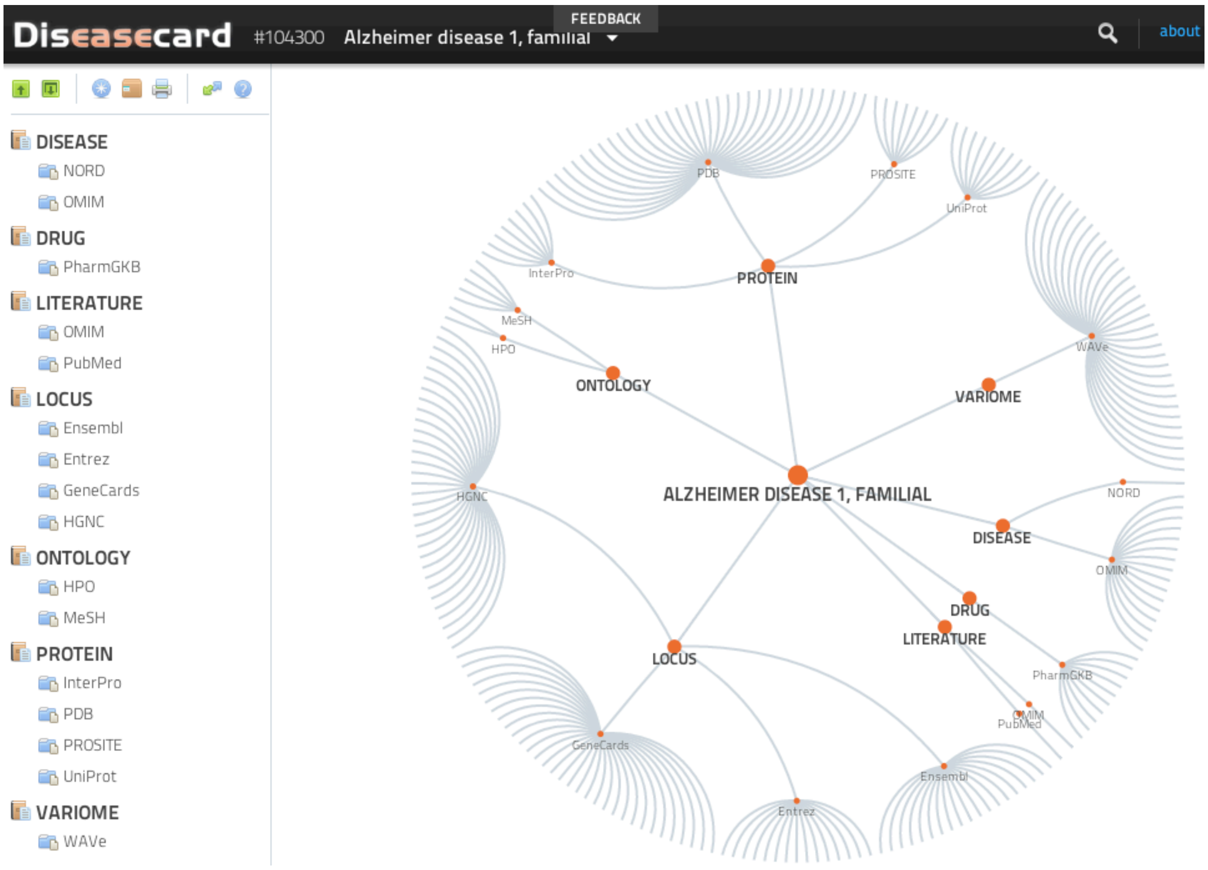


Figure 7-6. Diseasecard's entry navigation graph for Alzheimer's Disease, OMIM code 104300. A circular navigation map was created, using the JIT visualization library, to facilitate the access to the huge amount of linked resources.

64 http://thejit.org/

The navigation map starts with the selected disease and connections to the set of entities in the knowledge base. Clicking each entity name, *Protein* for example, centres the map on the *Protein* node, highlighting its connections to its various internal concepts. Likewise, clicking on a concept, *UniProt* for instance, centres the node and shows links to each individual concept Item.

Rich Data

Selecting the adequate set of resources for phenotype-oriented information was a crucial step towards the new Diseasecard. As expected, each resource features its own domain, architecture and interface standards, i.e., resources are heterogeneous and distributed. Table 7-2 list the resources and pointers integrated in the new Diseasecard.

Table 7-2. Diseasecard integrated resources.

NODE	RESOURCE	DESCRIPTION
Disease	NORD	http://www.rarediseases.org/
	OMIM	http://www.omim.org/
Drug	PharmGKB	http://www.pharmgkb.org/
Literature	Pubmed	http://www.ncbi.nlm.nih.gov/pubmed/
Locus	Ensembl	http://www.ensembl.org/
	Entrez	http://www.ncbi.nlm.nih.gov/gene/
	GeneCards	http://www.genecards.org/
	HGNC	http://www.genenames.org/
Ontology	GO	http://amigo.geneontology.org/
Protein	UniProt/SwissProt	http://www.uniprot.org/
	UniProt/TrEMBL	http://www.uniprot.org/
	PDB	http://www.pdb.org/
	Expasy	http://expasy.org/
	InterPro	http://www.ebi.ac.uk/interpro/
Variome	WAVe	http://bioinformatics.ua.pt/WAVe

One of COEUS major features is the prompt availability of interoperability services. In Diseasecard, the services required for accessing data are enabled by default.

The wealth of data collected during the data integration process is available through REST services, a SPARQL endpoint and a LinkedData view. Furthermore, the REST services and SPARQL endpoint are already used within Diseasecard client-side application.

Considering the constructed knowledge base a single platform without Diseasecard's web application, it is, *per se,* a single unique resource for the rare disease community. Life sciences developers can exploit this data collection to build or extend existing applications.

From a modern application perspective, Diseasecard can also be seen as platform with multiple applications. While for now only a web information system is available, COEUS robustness permits deploying applications targeted at the desktop or mobile devices, using the same set of APIs and accessing the original knowledge base.

The rare diseases dataset build using COEUS flexible integration engine results in a strikingly rich semantic knowledge base. This opens the room for further exploratory endeavours, namely using reasoning and inference. Whilst these features are not yet a part of the new Diseasecard, they will be made available through innovative user interactions in Diseasecard's web workspace.

7.3 Discussion

7.3.1 A Suitable Software Infrastructure for each Bioinformatician

As the miscellaneous "omics" fields branch new domains and research specializations, the technological needs for each field revolve around a common set of problems. Managing data, accessing and integrating information from other laboratories, or providing recently discovered knowledge to others are essential steps in the path of making science.

While it is nearly impossible to satisfy the requirements from all life sciences research fields, we can promote the use of technologies and tools that facilitate accomplishing all of the project's software-related goals. In a broad sense, this is our main objective with the COEUS framework.

The COEUS platform is a powerful development environment. Its scalability and flexibility make it ideal for highly heterogeneous scenarios and apt for the challenging requirements associated with particular "omics" fields. In fact, COEUS targets the improvement of niche fields, empowering amateur and professional developers with the tools to quickly model, integrate and publish data. Furthermore, the semantic web ideals span through all framework's components, from the integration of data to the interoperability with other tools. This provides limitless resource integration architectures with advanced data exploration features. With the former we are able to triplify almost all existing data into a new richer knowledge base and, complementarily, with the latter, we are able to access collected data by multiple means. This enables the creation of custom application ecosystems with comprehensive architectures. Bioinformaticians can program interfaces in any language and easily target web, desktop or mobile environments. Furthermore, by publishing data through the SPARQL and LinkedData interfaces, new systems will be part of the global web of knowledge. With the widespread use of COEUS and other similar tools, we expect that each bioinformatician can create its own technological infrastructure that goes beyond the boundaries of project-specific internal use.

Whereas the strategies adopted in the EU-ADR Web Platform and WAVe consisted in the adoption of traditional relational databases, COEUS empowers the creation of semantic knowledge bases. This enables a whole new level of knowledge exploration through the aforementioned SPARQL and LinkedData interfaces that allow the creation of complex knowledge reasoning and inference features. Diseasecard's dataset has innate semantics; collected data has an undisclosed meaning that can be explored to obtain new vital connections between genes and diseases. Likewise, COEUS brings this semantic web layer, and its underlying semantic features, to all bioinformaticians in any life sciences field.

7.3.2 Applying COEUS to the Rare Diseases Research Field

Research on rare diseases is of growing importance in the last couple of years. Uncovering the underlying genetic causes of rare diseases is the first step towards a better comprehension of our health, making us one step closer of the individualized healthcare panacea. Moreover, the funding and interest in large-scale rare disease projects has been renewed, namely within the European Union.

Since 2004, Diseasecard has been contributing to this research field by providing a portal with information regarding rare diseases and connections to a myriad of resources contextualized to each disease. Despite its quality, the legacy Diseasecard is an out-dated system, with an architecture that is no longer efficient for the current bioinformatics landscape. With COEUS, we have the opportunity to overcome the original Diseasecard's caveats, deploying a richer application, with a reengineered architecture and re-designed user interface. As previously mentioned, the combination of rapid application development with biomedical semantics is a key enabler of the next-generation of bioinformatics and the new Diseasecard is the first step towards this bright future.

The new Diseasecard, powered by COEUS, improves on the legacy version in three distinct aspects, discussed next.

- With a **semantic knowledge base** supporting the application, collected and connected data are richer and more meaningful. Whilst these capabilities are not yet fully explored, the open possibilities are immense. Establishing new relationships between OMIM's disease data and multiple ontologies or external resources generates a comprehensive rare disease dataset, enabling the inference of unique connections that would not be possible to obtain otherwise.
- Data in the knowledge base are now **interoperable** using COEUS default API. REST services, a SPARQL endpoint and a LinkedData interface are available for others to explore the dense rare disease data graph compiled in the new Diseasecard.
- The new Diseasecard **web workspace** is a significant improvement over the legacy version. Disease synonyms, the navigation tree and the new navigation graph present a more interactive and usable interface.

Diseasecard is the first COEUS instance and represents our initial endeavour towards the future of web-based biomedical applications. The new Diseasecard is available online at http://bioinformatics.ua.pt/dc4/.

8. FUTURE PERSPECTIVES AND CONCLUSIONS

"When it comes to the future, there are three kinds of people: those who let it happen, those who make it happen, and those who wonder what happened."
- John M. Richardson, Jr.

This thesis is set against the backdrop of a number of challenges to effectively enhance the use of service composition strategies within the biomedical software domain. The most prominent of these issues regards the explosive growth of data size and complexity generated in large-scale life sciences research projects, smaller laboratories or general practitioners databases. This evolution is pushing forward new demands regarding the scope expansion from multiple "omics" branches to holistic systems biology visions. Addressing these issues, whilst bridging the gap from data acquisition to data publishing, is a huge problem for the bioinformatics domain, whose last decade was characterized by unrestricted developments. In parallel, other drawbacks such as the discovery of data, the semantic exploration of knowledge or the requirements from policy makers allows us to encompass and organize these demands in the area of service composition for integration and interoperability.

The work detailed in the preceding six chapters reports our efforts to understand service composition in the biomedical software domain and to introduce new solutions that leverage on innovative strategies to enhance the process of developing biomedical applications. This final chapter highlights the produced results, introduces future lines of work in various research and business areas, and discusses the open path for the next generation of bioinformatics software.

8.1 Results

The overarching objectives of this doctorate work revolved around three general goals: the **evaluation** of existing service composition strategies for biomedical applications, the **exploration** of state-of-the-art service composition technologies for use in the bioinformatics research domain, and the **development** of innovative service composition solutions to tackle the broad set of challenges behind the bioinformatics revolution.

Chapter 2 includes a detailed analysis of the biomedical applications domain. From this evaluation it is clear that the bioinformatics software landscape is fragmented and highly heterogeneous. Despite multiple efforts towards the standardization of data models and services, the uncanny relationships amongst these elements highlight the demand for innovative solutions.

Advanced service composition strategies are required to enable bioinformatics for the 21st century. Adequate integration and interoperability methods must be created and applied to assure the continuous evolution of this field, thus making the personalized medicine panacea a reality.

From the deep exploration of state-of-the-art strategies and technologies for service composition, discussed in chapter 3, three ideas arise to overcome biomedical software challenges: the use of rapid application development methods, the emergence of platform-centric architectures, and the semantic web paradigm.

Rapid application development ideals define a set of principles to foster the faster creation and re-creation of applications from scratch, aiming at a speedier software delivery rate. Platform-centric architectures are an emerging trend with growing relevance. Focused on a centralized knowledge base accessible through multiple APIs, this model is being used in major modern systems, from Evernote to GitHub. With this model validated in large-scale real-world scenarios, it must be moved to the biomedical software domain. At last, the semantic web paradigm, with its standards for describing, managing and exploring knowledge, is the perfect solution for fundamental life sciences challenges. Content heterogeneity, information distribution, data integration, software interoperability or reasoning and inference issues, among others, can be solved with the adoption of semantic web technologies and guidelines.

As a consequence of these evaluation and exploratory research, new strategies were developed to endow the life sciences community with innovative technologies for service composition in biomedical applications: the EU-ADR Web Platform, WAVe and COEUS. These contributions add real value to the service composition for integration and interoperability state-of-the-art.

The EU-ADR Web Platform introduces four major scientific advances for the research community, namely on the field of software interoperability:

→ a new interoperability standard was created to empower the interdisciplinary communication amongst the various EU-ADR project services;

→ a strategy based on Taverna workflows was devised to enable the combination of results from the various EU-ADR project services;

→ a strategy for a new Taverna-based workflow execution engine was implemented to facilitate the inclusion of Taverna workflows into generic Java applications;

→ a web-based workspace was designed to deliver advanced pharmacovigilance studies to any stakeholder in pharmacogenomics and drug safety.

WAVe is an unique resource for human variome data, supported by a set of new service composition strategies for resource integration:

→ the lightweight data integration strategy enables the easy gathering of data from miscellaneous heterogeneous resources;

→ the extensible data model permits the dynamic addition of new data relationships without breaking the application setup;

→ the REST API is a unique resource for querying genetic variation datasets;

→ the web interface introduces new features essential for in-context gene analysis and to ensure content accreditation.

Crossing the boundaries of integration and interoperability is COEUS. Combining the three identified strategies for groundbreaking service composition, our efforts were concerted towards conceiving innovative approaches to support a new semantic web rapid application development framework, sustained by a platform-centric application architecture.

The resulting framework, COEUS, encloses four essential breakthroughs for the biomedical software and semantic web communities:

→ the set of new algorithms created to enhance rapid application development, further fostering the adoption of semantic web ;

→ the flexible integration strategies designed and implemented to enable the semantic acquisition of data from heterogeneous resources;

→ the semantic web strategies envisaged to permit future-proof software interoperability and facilitate reasoning and inference over acquired knowledge;

→ the innovative approaches for distributed knowledge federation.

These results highlight the successful accomplishment of this thesis' aims, effectively replying to the initial research challenge.

8.2 Future Perspectives

8.2.1 Beyond Service Composition

With computer science evolution we changed the way we evaluate and use web services. A couple decades ago, thinking about services was directly related with intelligent agents. Then came the understanding of web services as essential software architecture elements, optimal for event-driven scenarios or large distributed ICT infrastructures. These approaches leveraged new requirements and the standardization of web services appeared naturally. Nowadays, this legacy use of web services is in decay.

When standards appeared, their imposed limitations were seen as beneficial for the progression of service composition strategies. The reasoning behind SOAP, WSDL and UDDI was clearly focused on controlling all interactions and data exchanges between a set of services. These constraints increase the system architecture robustness but severely hinder its flexibility and efficiency. This is the major cause for REST's success. Being based strictly on the open HTTP protocol, REST is becoming increasingly popular amongst developers mostly due to its unrestricted approach. Using REST, the data model validation is passed on to the application level where it can be controlled more freely. Notwithstanding, SOAP is still the best solution for tightly controlled scenarios, as shown with the EU-ADR Web Platform.

Without the format and standard limitations, REST enables the exchanges of much richer data, at both ends of the spectrum. On the one hand, we can easily exchange heavy RDF data represented in any format and adhering to any ontology. On the other hand, we can exchange light JSON objects through really fast APIs. Hence, the adoption of REST guidelines in both WAVe and COEUS.

Within the various strategies developed in this thesis' work, the state-of-the-art is covered and future enhancements to this field are proposed. The strategies supporting the EU-ADR Web Platform, WAVe and COEUS actively go beyond service composition, delivering advanced algorithms for data integration and software interoperability.

The future withholds many developments in cloud computing and these architectures are the pinnacle of software-as-a-service. For instance, in Amazon's offer, everything can be consumed as a service, from data storage to processing power. With these cloud-centric approaches, data are always online and always available through and for miscellaneous web services. Whether we are posting something on Twitter or performing BLAST sequence alignments, web services will be intrinsically and transparently used in these operations. With the combination of cloud-based architectures and software-as-a-service ideals, the future of research goes beyond simple service compositions to completely new reproducible research environments, where data, features and services act as one entity available to everybody.

8.2.2 Linked Data and the Semantic Web

The future of the Internet lies in its evolution to more semantic environments. The ongoing transition from a Web of documents to a Web of data will lead to a new Internet, where anyone can easily query and reason over data (and its inner relationships) that are distributed and disjoint by nature. With current technologies we can already implement systems that automatically generate service composition workflows based on a set of rules predefined in an ontology. Internet 3.0, the intelligent Internet is already here.

The Linked Data Cloud is growing exponentially with data from the most diverse disciplines. The W3C recently approved a new interest group focused on understanding and promoting the use of Linked Data as a platform[65]. Moreover, semantic features are already intrinsically inside everyday tools, such as Google[66] or Facebook[67]. In addition, the power of semantics is also being harvested for applications in the industry for mobile scenarios, health care environments or media campaigns.

While the life sciences are definitive innovation driver in this field, there are challenges ahead to fill in the gaps surrounding this immense technological potential. COEUS is a suitable tool to tackle these challenges, with its combination of semantic web ideals and rapid application development principles. To build the Internet as a worldwide knowledge network will require efforts from all stakeholders involved in the life sciences community and, to be a part of this network, bioinformatics developers must embrace the change to a new semantic web paradigm, a change actively facilitated by COEUS.

[65] http://www.w3.org/2012/ldp/
[66] http://www.google.com/insidesearch/features/search/knowledge.html
[67] https://developers.facebook.com/docs/opengraph/

8.2.3 Worldwide Knowledge Networks

With a more intelligent Internet, we can explore and reason over data that are scattered everywhere. With data represented according to LinkedData guidelines, it will be easier to access data without a database: the Internet is the actual knowledge base. The new Web of Data will be the source for miscellaneous semantic integration systems that are capable of accurately understand and interpret heterogeneous data whilst hiding the complexities of the underlying semantic technologies.

Nowadays, most available tools are highly sophisticated platforms completely out of the general developers' reach. With new frameworks like COEUS, the transitions from a legacy environment to the modern Web of Data are eased. To be a part of this worldwide knowledge network, existing data must be translated into new semantic formats. Work on data triplification enables the production of rich knowledge bases. Continuing this effort to create semantic versions that replace existing resources is essential to reveal the true knowledge behind them, through advanced reasoning and knowledge inference strategies. Even if the adoption of these strategies is only tangible and focused on niche fields, there are already technologies to connect these highly specialized systems to the Web of Data, making them an active part of a truly intelligent, worldwide, distributed, knowledge network.

The data behind this network foster the creation of more modern software platforms, using richer datasets and, consequently, providing richer user features. The most recent success case is the 2012 Olympics sports site built by BBC[68]. The BBC's sports knowledge base contains linked information for all athletes, teams, nations and modalities, among others, occurring during the event. This huge semantic knowledge base powers up the entire news infrastructure. In addition to allowing reporters to write richer news, BBC's architecture allows the dynamic creation of semantically rich applications, targeting mobile or web environments, containing all the relevant information for a given topic or domain.

8.2.4 Modern Software Platforms

In the post-PC era mobile devices are gaining relevance in our daily lives and work. Smartphones and tablets are no longer simple communication devices; they are an open door to the information highway that can be use for learning, leisure or work. Along with

[68] http://www.bbc.co.uk/blogs/bbcinternet/2012/04/sports_dynamic_semantic.html

the evolution of our conception of the personal computer came the democratization of technology. Everyone is familiar with a computer and, most notably, the software development process is now fairly trivial. With the latter came a new wave of technologies and frameworks that allow almost anyone to build a new application. Likewise, this also prompted many experts from miscellaneous domains to start developing software. The outcome of this evolution can be assessed in two ways. On the one hand, there are many more high-quality applications using all the richness provided by our devices. However, on the other hand, the overall quality standard of the applications has also been lowered.

New applications are built from scratch to interact with multiple ecosystems or to create new ones. We cannot imagine any developer launching a social application without including Facebook connections. Similarly, crafting new user and data ecosystems can leverage on existing semantics to provide deeper and meaningful interactions with other networks. Whether we are building a new protein interaction network or a social network, the relationship semantics and the demands to interact with external environments are omnipresent.

With virtually centralized cloud-based data stores and software-as-a-service ideals, the door is open for rapid application development frameworks. Creating a modern software platform is nowadays a task of "code once, build to many" where the service-based implementations are shared through desktop, web, TV and mobile clients. COEUS underlying architecture enables the adoption of this trend by bioinformatics developers, tackling the challenge of moving innovative technologies to the life sciences. The principles and ideals behind modern software architectures are common to bioinformatics systems and the overall life sciences community, from wet-lab researchers to clinicians, will be tremendously benefited from embracing them.

8.2.5 Business Value

Nowadays, research and enterprise can no longer afford to live in separate worlds. Research requires private funding and interest to be sustainable and enterprises rely on advanced research to continue the pursuit of innovation. For this matter, it is our belief that every research work should be envisaged as a commercial product. From the various contributions discussed in this thesis, the COEUS framework is the one with the biggest business value.

COEUS tackles not only the challenges regarding the use of semantic web technologies, but also the ones inherent to deploying a modern software platform. Whilst the first case study, Diseasecard, is built for the biomedical applications domain, many other scenarios can be explored with COEUS. The comprehensive integration connectors, the flexible triplification engine and the set of data exploration APIs, make COEUS suitable for adoption in pharmaceutical industry solutions, news aggregations, multimedia datasets or education.

From a marketing perspective, the advanced client-side development features allow the creation of multiple applications sharing the same codebase. In a sense, COEUS permits the creation of our own custom data and services cloud, enabling the introduction of several applications tailored to different market segments. With COEUS, the gap from research to business is actively reduced, allowing future endeavours in building a commercial product.

8.3 Conclusion

The time has come to take bioinformatics software development to a whole new level. Computer sciences' advances over the last decade have launched a series of transitions in the way we understand and use technologies. Service composition strategies have matured and play a vital role in the evolution of modern software platforms. In parallel, the quest for knowledge is also improved by the emergence of the semantic web paradigm. Stemming from artificial intelligence, the semantic web principles are revolutionizing the way we store, share, reason and explore available data. All these innovative technological advances must find their way into biomedical applications.

The various solutions reported in this thesis are a remarkable evolution over the current practices for service composition strategies in biomedical software, especially the new proposed methods that better suit bioinformatics' demands. In this field, service composition revolves around integration and interoperability. This means that we must identify the best algorithms to combine the many pieces for the integration of resources or for the publication of the content in a knowledge base through interoperable interfaces. Where WAVe is focused on the former and the EU-ADR Web Platform on the latter, COEUS covers both ends with a unique approach to combine semantic web technologies with the need for integration and interoperability. The practicality of COEUS resides in its streamlined development process. By adopting rapid application development principles, the framework offers control over a set of reusable assets that can be quickly mixed to build modern, dynamic and intelligent software ecosystems.

The combination of all contributions described in this thesis accomplishes the initial objectives. We have individually assessed multiple service composition strategies for integration and interoperability and created products using new innovative solutions. Moreover, we have successfully introduced a new software package that goes beyond what has been done before, effectively fulfilling this doctorate research goals and contributing to the advance of the biomedical applications state-of-the-art.

REFERENCES

[1] R. Sachidanandam, D. Weissman, *et al.*, "A map of human genome sequence variation containing 1.42 million single nucleotide polymorphisms", *Nature,* vol. 409, pp. 928-933, 2001. [doi:10.1038/35057149]

[2] J. C. Venter, M. D. Adams, *et al.*, "The sequence of the human genome", *Science's STKE,* vol. 291, p. 1304, 2001.

[3] F. S. Collins, E. D. Green, *et al.*, "A vision for the future of genomics research", *Nature,* vol. 422, pp. 835-847, 2003.

[4] D. Cortese, "A vision of individualized medicine in the context of global health", *Clinical Pharmacology & Therapeutics,* vol. 82, pp. 491-493, 2007.

[5] C. Sander, "Genomic medicine and the future of health care", *Science,* vol. 287, pp. 1977-1978, 2000.

[6] A. E. Guttmacher and F. S. Collins, "Realizing the promise of genomics in biomedical research", *JAMA: The Journal of the American Medical Association,* vol. 294, pp. 1399-1402, 2005.

[7] B. Louie, P. Mork, *et al.*, "Data integration and genomic medicine", *Journal of Biomedical Informatics,* vol. 40, pp. 5-16, 2007.

[8] P. Lopes, J. Arrais, and J. L. Oliveira, "Dynamic Service Integration using Web-based Workflows", in *10th International Conference on Information Integration and Web Applications & Services*, Linz, Austria, 2008, pp. 622-625. [doi:10.1145/1497308.1497426]

[9] P. Lopes, J. Arrais, and J. Oliveira, "DynamicFlow: A Client-Side Workflow Management System", in *Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing, and Ambient Assisted Living*, 2009, pp. 1101-1108.

[10] P. Lopes and J. L. Oliveira, "An extensible platform for variome data integration", in *10th IEEE International Conference on Information Technology and Applications in Biomedicine (ITAB)*, Corfu, Greece, 2010, pp. 1-4. [doi:10.1109/ITAB.2010.5687784]

[11] P. Lopes, R. Dalgleish, and J. L. Oliveira, "WAVe: Web Analysis of the Variome", *Human Mutation,* vol. 32, Mar 10 2011. [doi:10.1002/humu.21499]

[12] P. Lopes and J. L. Oliveira, "Towards knowledge federation in biomedical applications", in *7th International Conference on Semantic Systems (I-SEMANTICS)*, Graz, Austria, 2011, pp. 87-94. [doi:10.1145/2063518.2063530]

[13] P. Lopes, José, and L. Oliveira, "A semantic web application framework for health systems interoperability", in *First International Workshop on Managing Interoperability and Complexity in Health Systems (MIX-HS)*, Glasgow, Scotland, UK, 2011, pp. 87-90. [doi:10.1145/2064747.2064768]

[14] P. Lopes and J. L. Oliveira, "COEUS: A Semantic Web Application Framework", in *4th International Workshop on Semantic Web Applications and Tools for the Life Sciences (SWAT4LS)*, London, United Kingdom, 2012, pp. 66-73. [doi:10.1145/2166896.2166915]

[15] J. D. Watson, "The human genome project: past, present, and future", *Science,* vol. 248, pp. 44-49, April 6, 1990 1990. [doi:10.1126/science.2181665]

[16] R. Tupler, G. Perini, and M. R. Green, "Expressing the human genome", *Nature,* vol. 409, pp. 832-833, 2001.

[17] D. Primorac, "Human Genome Project-based Applications in Forensic Science, Anthropology, and Individualized Medicine", *Croat Med J,* vol. 50, pp. 205-6, Jun 2009 2009. [doi:10.3325/cmj.2009.50.205]

[18] L. G. Biesecker, J. C. Mullikin, *et al.*, "The ClinSeq Project: Piloting large-scale genome sequencing for research in genomic medicine", *Genome Research,* vol. 19, pp. 1665-1674, 2009. [doi:10.1101/gr.092841.109]

[19] H. Z. Ring, P.-Y. Kwok, and R. G. Cotton, "Human Variome Project: an international collaboration to catalogue human genetic variation", *Pharmacogenomics,* vol. 7, pp. 969-972, 2006. [doi:doi:10.2217/14622416.7.7.969]

[20] B. Giardine, C. Riemer, *et al.*, "PhenCode: connecting ENCODE data with mutations and phenotype", *Human Mutation,* vol. 28, pp. 554-562, 2007.

[21] G. Trifiro, A. Fourrier-Reglat, *et al.*, "The EU-ADR project: preliminary results and perspective", *Studies In Health Technology And Informatics,* vol. 148, pp. 43-49, 2009.

[22] D. E. Reich and D. B. Goldstein, "Genetic evidence for a Paleolithic human population expansion in Africa", *Proceedings of the National Academy of Sciences,* vol. 95, pp. 8119-8123, July 7, 1998 1998.

[23] L. Kruglyak and D. A. Nickerson, "Variation is the spice of life", *Nature Genetics,* vol. 27, pp. 234-236, 2001.

[24] R. A. Gibbs, J. W. Belmont, *et al.*, "The international HapMap project", *Nature,* vol. 426, pp. 789-796, 2003.

[25] G. A. Thorisson, A. V. Smith, *et al.*, "The international HapMap project web site", *Genome Research,* vol. 15, pp. 1592-1593, 2005.

[26] M. Via, C. Gignoux, and E. G. Burchard, "The 1000 Genomes Project: new opportunities for research and social challenges", *Genome Medicine,* vol. 2, pp. 1-3, 2010.

[27] A. Hamosh, A. F. Scott, *et al.*, "Online Mendelian Inheritance in Man (OMIM), a knowledgebase of human genes and genetic disorders", *Nucleic Acids Research,* vol. 33, pp. D514-D517, January 1, 2005 2005. [doi:10.1093/nar/gki033]

[28] J. R. Riordan, J. M. Rommens, *et al.*, "Identification of the cystic fibrosis gene: cloning and characterization of complementary DNA", *Science,* vol. 245, pp. 1066-1073, 1989.

[29] B. Kerem, J. M. Rommens, *et al.*, "Identification of the cystic fibrosis gene: genetic analysis", *Science,* vol. 245, pp. 1073-1080, 1989.

[30] L. L. Cavalli-Sforza and M. W. Feldman, "The application of molecular genetic approaches to the study of human evolution", *Nature Genetics,* vol. 33, pp. 266-275, 2003.

[31] M. Y. Galperin and X. M. Fernández-Suárez, "The 2012 Nucleic Acids Research Database Issue and the online Molecular Biology Database Collection", *Nucleic Acids Research,* vol. 40, pp. D1-D8, January 1, 2012 2012. [doi:10.1093/nar/gkr1196]

[32] S. T. Sherry, M.-H. Ward, *et al.*, "dbSNP: the NCBI database of genetic variation", *Nucleic Acids Research,* vol. 29, pp. 308-311, January 1, 2001 2001. [doi:10.1093/nar/29.1.308]

[33] C. E. Lipscomb, "Medical Subject Headings (MeSH)", *Bulletin of the Medical Library Association,* vol. 88, pp. 265-6, Jul 2000.

[34] D. A. Benson, I. Karsch-Mizrachi, *et al.*, "GenBank", *Nucleic Acids Research,* vol. 38, pp. D46-51, January 1, 2010 2010. [doi:10.1093/nar/gkp1024]

[35] D. Maglott, J. Ostell, *et al.*, "Entrez Gene: gene-centered information at NCBI", *Nucleic Acids Research,* vol. 35, 2007.

[36] A. Bairoch, R. Apweiler, *et al.*, "The Universal Protein Resource (UniProt)", *Nucleic Acids Research,* vol. 33, pp. D154-159, January 1, 2005 2005. [doi:10.1093/nar/gki070]

[37] N. J. Mulder, R. Apweiler, *et al.*, "New developments in the InterPro database", *Nucleic Acids Research,* vol. 35, pp. D224-228, January 12, 2007 2007. [doi:10.1093/nar/gkl841]

[38] E. Gasteiger, A. Gattiker, *et al.*, "ExPASy: the proteomics server for in-depth protein knowledge and analysis", *Nucleic Acids Research,* vol. 31, pp. 3784-3788, 2003.

[39] C. J. A. Sigrist, L. Cerutti, *et al.*, "PROSITE, a protein domain database for functional characterization and annotation", *Nucleic Acids Research,* vol. 38, pp. D161-D166, 2010.

[40] H. Parkinson, U. Sarkans, *et al.*, "ArrayExpress--a public repository for microarray gene expression data at the EBI", *Nucleic Acids Research,* vol. 33, pp. D553-555, January 1, 2005 2005. [doi:10.1093/nar/gki056]

[41] T. Margaria, M. G. Hinchey, *et al.*, "Ensembl Database: Completing and Adapting Models of Biological Processes", *Proceedings of the Conference on Biologically Inspired Cooperative Computing (BiCC IFIP): 20-25 August 2006; Santiago (Chile),* pp. 43 - 54, 2006.

[42] M. Kanehisa and S. Goto, "KEGG: Kyoto Encyclopedia of Genes and Genomes", *Nucleic Acids Research,* vol. 28, pp. 27-30, January 1, 2000 2000. [doi:10.1093/nar/28.1.27]

[43] M. Kanehisa, S. Goto, *et al.*, "From genomics to chemical genomics: new developments in KEGG", *Nucleic Acids Research,* vol. 34, pp. D354-357, January 1, 2006 2006. [doi:10.1093/nar/gkj102]

[44] M. Ashburner, C. A. Ball, *et al.*, "Gene Ontology: tool for the unification of biology", *Nature Genetics,* vol. 25, pp. 25 - 9, 2000.

[45] R. Stevens, C. A. Goble, and S. Bechhofer, "Ontology-based knowledge representation for bioinformatics", *Briefings in Bioinformatics,* vol. 1, pp. 398-414, January 1, 2000 2000. [doi:10.1093/bib/1.4.398]

[46] Y. Lussier, T. Borlawsky, *et al.*, "PhenoGO: assigning phenotypic context to gene ontology annotations with natural language processing", *Pac Symp Biocomput,* pp. 64-75, 2006.

[47] L. Sam, E. Mendonca, *et al.*, "PhenoGO: an integrated resource for the multiscale mining of clinical and biological data", *BMC Bioinformatics,* vol. 10, p. S8, 2009.

[48] A. Kahraman, A. Avramov, *et al.*, "PhenomicDB: a multi-species genotype/phenotype database for comparative phenomics", *Bioinformatics,* vol. 21, pp. 418-420, February 1, 2005 2005. [doi:10.1093/bioinformatics/bti010]

[49] P. Groth, N. Pavlova, *et al.*, "PhenomicDB: a new cross-species genotype/phenotype resource", *Nucleic Acids Research,* vol. 35, pp. D696-699, January 12, 2007 2007. [doi:10.1093/nar/gkl662]
[50] C. F. Thorn, T. E. Klein, and R. B. Altman, "Pharmacogenomics and bioinformatics: PharmGKB", *Pharmacogenomics,* vol. 11, pp. 501-505, 2010.
[51] G. A. Thorisson, O. Lancaster, *et al.*, "HGVbaseG2P: a central genetic association database", *Nucleic Acids Research,* vol. 37, pp. D797-802, January 1, 2009 2009. [doi:10.1093/nar/gkn748]
[52] E. Neumann, "A life science Semantic Web: are we there yet?", *Science's STKE,* vol. 2005, p. pe22, 2005.
[53] B. M. Good and M. D. Wilkinson, "The life sciences semantic web is full of creeps!", *Briefings in Bioinformatics,* vol. 7, pp. 275-286, 2006.
[54] A. Ruttenberg, J. A. Rees, *et al.*, "Life sciences on the Semantic Web: the Neurocommons and beyond", *Briefings in Bioinformatics,* vol. 10, pp. 193-204, 2009.
[55] F. Belleau, M.-A. Nolin, *et al.*, "Bio2RDF: Towards a mashup to build bioinformatics knowledge systems", *Journal of Biomedical Informatics,* vol. 41, pp. 706-716, 2008.
[56] N. F. Noy, N. H. Shah, *et al.*, "BioPortal: ontologies and integrated data resources at the click of a mouse", *Nucleic Acids Research,* vol. 37, pp. W170-W173, 2009.
[57] P. L. Whetzel, N. F. Noy, *et al.*, "BioPortal: enhanced functionality via new Web services from the National Center for Biomedical Ontology to access and use ontologies in software applications", *Nucleic Acids Research,* vol. 39, pp. W541-W545, 2011. [doi:10.1093/nar/gkr469]
[58] T. Berners-Lee, "Linked data-design issues", *W3C,* vol. 2009, 2006.
[59] C. Bizer, "The Emerging Web of Linked Data", *Intelligent Systems, IEEE,* vol. 24, pp. 87-92, 2009. [doi:10.1109/mis.2009.102]
[60] C. Bizer, T. Heath, and T. Berners-Lee, "Linked Data - The Story So Far", ed: IGI Global, 2009, pp. 1-22. [doi:10.4018/jswis.2009081901]
[61] C. Bizer, J. Lehmann, *et al.*, "DBpedia - A crystallization point for the Web of Data", *Web Semantics: Science, Services and Agents on the World Wide Web,* vol. 7, pp. 154-165, 2009. [doi:10.1016/j.websem.2009.07.002]
[62] A. Jenkinson, M. Albrecht, *et al.*, "Integrating biological data - the Distributed Annotation System", *BMC Bioinformatics,* vol. 9, p. S3, 2008.
[63] D. Smedley, S. Haider, *et al.*, "BioMart - biological queries made easy", *BMC Genomics,* vol. 10, p. 22, 2009.
[64] S. Haider, B. Ballester, *et al.*, "BioMart Central Portal--unified access to biological data", *Nucleic Acids Research,* vol. 37, pp. W23-27, July 1, 2009 2009. [doi:10.1093/nar/gkp265]
[65] P. Rice, I. Longden, and A. Bleasby, "EMBOSS: the European Molecular Biology Open Software Suite", *Trends Genet,* vol. 16, pp. 276-7, Jun 2000. [doi:S0168-9525(00)02024-2 [pii]]
[66] S. A. Olson, "EMBOSS opens up sequence analysis. European Molecular Biology Open Software Suite", *Briefings in Bioinformatics,* vol. 3, pp. 87-91, Mar 2002.
[67] M. Sarachu and M. Colet, "wEMBOSS: a web interface for EMBOSS", *Bioinformatics,* vol. 21, pp. 540-1, Feb 15 2005. [doi:10.1093/bioinformatics/bti031]
[68] S. Pillai, V. Silventoinen, *et al.*, "SOAP-based services provided by the European Bioinformatics Institute", *Nucleic Acids Research,* vol. 33, pp. W25-28, July 1, 2005 2005. [doi:10.1093/nar/gki491]
[69] M. Wilkinson and M. Links, "BioMoby: An open source biological web services proposal", *Briefings in Bioinformatics,* vol. 3, pp. 331 - 341, 2002.
[70] M. DiBernardo, R. Pottinger, and M. Wilkinson, "Semi-automatic web service composition for the life sciences using the BioMoby semantic web framework", *Journal of Biomedical Informatics,* vol. 41, pp. 837-847, 2008.
[71] H. Sugawara and S. Miyazaki, "Biological SOAP servers and web services provided by the public sequence data bank", *Nucleic Acids Research,* vol. 31, pp. 3836-3839, July 1, 2003 2003. [doi:10.1093/nar/gkg558]
[72] Y. Kwon, Y. Shigemoto, *et al.*, "Web API for biology with a workflow navigation system", *Nucleic Acids Research,* vol. 37, pp. W11-16, July 1, 2009 2009. [doi:10.1093/nar/gkp300]
[73] J. Bhagat, F. Tanoh, *et al.*, "BioCatalogue: a universal catalogue of web services for the life sciences", *Nucleic Acids Research,* vol. 38, pp. W689-W694, July 1, 2010 2010. [doi:10.1093/nar/gkq394]
[74] C. Goble and R. Stevens, "State of the nation in data integration for bioinformatics", *Journal of Biomedical Informatics,* vol. 41, pp. 687-693, 2008. [doi:10.1016/j.jbi.2008.01.008]
[75] L. D. Stein, "Creating a Bioinformatics Nation", *Nature,* vol. 417, pp. 119 - 120, 2002.
[76] L. D. Stein, "Integrating biological databases", *Nature Genetics,* vol. 4, pp. 337-345, 2003.
[77] J. Arrais, B. Santos, *et al.*, "GeneBrowser: an approach for integration and functional classification of genomic data", *Journal of Integrative Bioinformatics,* vol. 4, 2007. [doi:10.2390/biecoll-jib-2007-82]

[78] J. Arrais, J. Fernandes, *et al.*, "GeneBrowser 2: an application to explore and identify common biological traits in a set of genes", *BMC Bioinformatics,* vol. 11, p. 389, 2010. [doi:10.1186/1471-2105-11-389]

[79] J. Arrais, J. Pereira, and J. L. Oliveira, "GeNS: A biological data integration platform", in *ICBB 2009, International Conference on Bioinformatics and Biomedicine*, Venice, 2009.

[80] A. Birkland and G. Yona, "BIOZON: a system for unification, management and analysis of heterogeneous biological data", *BMC Bioinformatics,* vol. 7, 2006.

[81] I. Vastrik, P. D'Eustachio, *et al.*, "Reactome: a knowledge base of biologic pathways and processes", *Genome Biolology,* vol. 8, p. R39, 2007.

[82] E. K. Neumann and D. Quan, "BioDASH: a Semantic Web dashboard for drug development", *Pac Symp Biocomput,* pp. 176 - 187, 2006.

[83] R. C. G. Holland, T. A. Down, *et al.*, "BioJava: an open-source framework for bioinformatics", *Bioinformatics,* vol. 24, pp. 2096-2097, 2008.

[84] N. Goto, P. Prins, *et al.*, "BioRuby: Bioinformatics software for the Ruby programming language", *Bioinformatics,* vol. 26, pp. 2617-2619, 2010.

[85] J. E. Stajich, D. Block, *et al.*, "The Bioperl toolkit: Perl modules for the life sciences", *Genome Research,* vol. 12, pp. 1611-1618, 2002.

[86] P. J. A. Cock, T. Antao, *et al.*, "Biopython: freely available Python tools for computational molecular biology and bioinformatics", *Bioinformatics,* vol. 25, pp. 1422-1423, 2009.

[87] T. Margaria, R. Nagel, and B. Steffen, "Bioconductor-jETI: A Tool for Remote Tool Integration", *Proceedings of the 11th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS): 4-8 April 2005; Edinburgh, U.K.,* pp. 557 - 562, 2005.

[88] M. Swertz, M. Dijkstra, *et al.*, "The MOLGENIS toolkit: rapid prototyping of biosoftware at the push of a button", *BMC Bioinformatics,* vol. 11, p. S12, 2010.

[89] R. Stevens, A. Robinson, and C. Goble, "myGrid: personalized bioinformatics on the information grid", *Bioinformatics,* vol. 19, pp. I302 - I304, 2003.

[90] M. A. Vouk, "Cloud computing - Issues, research and implementations", in *Information Technology Interfaces, 2008. ITI 2008. 30th International Conference on*, 2008, pp. 31-40.

[91] J. T. Dudley and A. J. Butte, "In silico research in the era of cloud computing", *Nat Biotech,* vol. 28, pp. 1181-1185, 2010.

[92] T. Margaria, C. Kubczak, and B. Steffen, "Bio-jETI: a service integration, design, and provisioning platform for orchestrated bioinformatics processes", *BMC Bioinformatics,* vol. 9, p. S12, 2008.

[93] E. Bartocci, F. Corradini, *et al.*, "BioWMS: a web-based Workflow Managemt System for bioinformatics", *BMC Bioinformatics,* vol. 8, p. 14, 2007.

[94] P. Romano, D. Marra, and L. Milanesi, "Web services and workflow management for biological resources", *BMC Bioinformatics,* vol. 6, p. S24, 2005.

[95] P. Romano, E. Bartocci, *et al.*, "Biowep: a workflow enactment portal for bioinformatic applications", *BMC Bioinformatics,* vol. 8, 2007.

[96] T. Life Sciences Practice, "BioWBI and WEE: Tools for Bioinformatics Analysis Workflows", 2004.

[97] Z. Guan and H. M. Jamil, "Streamlining biological data analysis using BioFlow", in *Third IEEE Symposium on Bioinformatics and Bioengineering*, 2003, pp. 258-262.

[98] H. Jamil and B. El-Hajj-Diab, "BioFlow: A Web-Based Declarative Workflow Language for Life Sciences", in *IEEE Congress on Services - Part I*, 2008, pp. 453 - 460. [doi:10.1109/SERVICES-1.2008.73]

[99] B. Ludascher, I. Altintas, *et al.*, "Taverna: Scientific Workflow Management and the Kepler System", *Research Articles, Concurrency and Computation: Practice & Experience,* vol. 18, pp. 1039 - 1065, 2006.

[100] B. Giardine, C. Riemer, *et al.*, "Galaxy: a platform for interactive large-scale genome analysis", *Genome Research,* vol. 15, pp. 1451-1455, 2005.

[101] J. Goecks, A. Nekrutenko, and J. Taylor, "Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences", *Genome Biology,* vol. 11, p. R86, 2010.

[102] M. Abouelhoda, S. Issa, and M. Ghanem, "Tavaxy: Integrating Taverna and Galaxy workflows with cloud computing support", *BMC Bioinformatics,* vol. 13, p. 77, 2012.

[103] P. Groth, A. Gibson, and J. Velterop, "The anatomy of a nanopublication", *Information Services and Use,* vol. 30, pp. 51-56, 2010. [doi:10.3233/isu-2010-0613]

[104] H. M. Sneed, "Software evolution. A road map", in *Software Maintenance, 2001. Proceedings. IEEE International Conference on*, 2001, p. 7.

[105] Z. Zou, Z. Duan, and J. Wang, "A Comprehensive Framework for Dynamic Web Services Integration", in *European Conference on Web Services (ECOWS'06)*, 2006.

[106] G. O. H. Chong Minsk, L. E. E. Siew Poh, *et al.*, "Web 2.0 Concepts and Technologies for Dynamic B2B Integration", *IEEE,* pp. 315-321, 2007.

[107] M. Turner, D. Budgen, and P. Brereton, "Turning software into a service", *Computer,* vol. 36, pp. 38-44, 2003.

[108] L. Xuanzhe, H. Yi*, et al.*, "Towards Service Composition Based on Mashup", in *IEEE Congress on Services*, 2007, pp. 332-339.

[109] N. Yan, "Build Your Mashup with Web Services", in *IEEE International Conference on Services Computing*, Salt Lake City, UT, USA, 2007. [doi:10.1109/SCC.2007.34]

[110] Q. Zhao, G. Huang*, et al.*, "A Web-Based Mashup Environment for On-the-Fly Service Composition", in *Service-Oriented System Engineering, 2008. SOSE '08. IEEE International Symposium on*, 2008, pp. 32-37.

[111] D. Hollingsworth, *The Workflow Reference Model*, 1995.

[112] G. Preuner and M. Schrefl, "Integration of Web Services into Workflows through a Multi-Level Schema Architecture", in *4th IEEE Int'l Workshop on Advanced Issues of E-Commerce and Web-Based Information Systems (WECWIS 2002)*, 2002.

[113] J. Cardoso and A. Sheth, "Semantic E-Workflow Composition", *Journal of Intelligent Information Systems,* 2003.

[114] P. C. K. Hung and D. K. W. Chiu, "Developing Workflow-based Information Integration (WII) with Exception Support in a Web Services Environment", in *37th Hawaii International Conference on System Sciences - 2004*, 2004.

[115] Y. Gil, E. Deelman*, et al.*, "Examining the Challenges of Scientific Workflows", *Computer,* vol. 40, pp. 24-32, 2007.

[116] T. McPhillips, S. Bowers*, et al.*, "Scientific workflow design for mere mortals", *Future Generation Computer Systems,* vol. 25, pp. 541-551, 2009. [doi:10.1016/j.future.2008.06.013]

[117] S. Petkov, E. Oren, and A. Haller. (2005). *Aspects in Workflow Management*.

[118] P. Beynon-Davies, C. Carne*, et al.*, "Rapid application development (RAD): an empirical review", *European Journal of Information Systems,* vol. 8, pp. 211-223, 1999.

[119] R. Agarwal, J. Prasad*, et al.*, "Risks of rapid application development", *Commun. ACM,* vol. 43, p. 1, 2000. [doi:10.1145/352515.352516]

[120] T. Inoue, H. Asakura*, et al.*, "Rapid Development of Web Applications by Introducing Database Systems with Web APIs", in *Database Systems for Advanced Applications*. vol. 5982, H. Kitagawa, Y. Ishikawa*, et al.*, Eds., ed: Springer Berlin / Heidelberg, 2010, pp. 327-336. [doi:10.1007/978-3-642-12098-5_27]

[121] C. Peltz, "Web services orchestration and choreography", *Computer,* vol. 36, pp. 46-52, 2003.

[122] S. Staab, "Web Services: Been there, Done That?", *IEEE Intelligent Systems,* pp. 72-85, 2003.

[123] W3C, "Web Services", ed: World Wide Web Consortium, 2002.

[124] W3C, "Simple Object Access Protocol", ed: World Wide Web Consortium, 2007.

[125] OASIS, "Universal Description, Discovery and Integration", ed: OASIS, 2005.

[126] W3C, "Web Service Description Language", ed: World Wide Web Consortium, 2001.

[127] J. Kopecky, T. Vitvar*, et al.*, "SAWSDL: Semantic Annotations for WSDL and XML Schema", *Internet Computing, IEEE,* vol. 11, pp. 60-67, 2007.

[128] R. T. Fielding and R. N. Taylor, "Principled design of the modern Web architecture", *ACM Trans. Internet Technol.,* vol. 2, pp. 115-150, 2002. [doi:10.1145/514183.514185]

[129] F. Rosenberg, F. Curbera*, et al.*, "Composing RESTful Services and Collaborative Workflows: A Lightweight Approach", *Internet Computing, IEEE,* vol. 12, pp. 24-31, 2008.

[130] S. S. S. Reddy, L. S. S. Reddy*, et al.*, "Advanced Techniques for Scientific Data Warehouses", in *International Conference on Advanced Computer Control, ICACC*, 2009, pp. 576-580.

[131] N. Polyzotis, S. Skiadopoulos*, et al.*, "Meshing Streaming Updates with Persistent Data in an Active Data Warehouse", *Knowledge and Data Engineering, IEEE Transactions on,* vol. 20, pp. 976-991, 2008.

[132] Y. Zhu, L. An, and S. Liu, "Data Updating and Query in Real-Time Data Warehouse System", in *Computer Science and Software Engineering, 2008 International Conference on*, 2008, pp. 1295-1297.

[133] A. Kiani and N. Shiri, "A Generalized Model for Mediator Based Information Integration", in *11th International Database Engineering and Applications Symposium*, 2007, pp. 268-272.

[134] L. M. Haas, P. M. Schwarz*, et al.*, "DiscoveryLink: A system for integrated access to life sciences data sources", *IBM Systems Journal,* vol. 40, pp. 489-511, 2001.

[135] G. Hohpe and B. Woolf, *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*: Addison-Wesley, 2004.

[136] R. Kazman, G. Abowd*, et al.*, "Scenario-based analysis of software architecture", *Software, IEEE,* vol. 13, pp. 47-55, 1996.

[137] A. Tolk and J. A. Muguira, "Levels of Conceptual Interoperability Model", in *Fall Simulation Interoperability Workshop*, Orlando, Florida, USA, 2003, pp. 14-19.

[138] T. Berners-Lee, J. Hendler, and O. Lassila, "The Semantic Web", *Sci Am,* vol. 284, pp. 34 - 43, 2001.

[139] M. Burstein, C. Bussler, *et al.*, "A semantic Web services architecture", *Internet Computing, IEEE,* vol. 9, pp. 72-81, 2005.

[140] M. Stollberg and A. Haller, "Semantic Web services tutorial", in *IEEE International Conference on Services Computing*, 2005. [doi:10.1109/SCC.2005.81]

[141] J. Hendler, "Enhanced: Science and the Semantic Web", *Science,* vol. 299, pp. 520-521, January 24, 2003 2003. [doi:10.1126/science.1078874]

[142] E. Neumann, "A Life Science Semantic Web: Are We There Yet?", *Science's STKE,* vol. 2005, pp. pe22-, May 10, 2005 2005. [doi:10.1126/stke.2832005pe22]

[143] L. Masinter, T. Berners-Lee, and R. T. Fielding, "Uniform resource identifier (URI): Generic syntax", 2005.

[144] W3C, "Resource Description Framework", ed: World Wide Web Consortium, 2004.

[145] W3C, "Web Ontology Language", ed: World Wide Web Consortium, 2007.

[146] S. Harris and N. Shadbolt, "SPARQL Query Processing with Conventional Relational Database Systems", in *Web Information Systems Engineering – WISE 2005 Workshops*, ed, 2005, pp. 235-244.

[147] A. Ruttenberg, T. Clark, *et al.*, "SPARQL: Advancing translational research with the Semantic Web", *BMC Bioinformatics,* vol. 8, p. S2, 2007.

[148] E. J. Miller, "An Introduction to the Resource Description Framework", *Journal of Library Administration,* vol. 34, pp. 245-255, 2001.

[149] M. Uschold and M. Gruninger, "Ontologies: Principles, Methods and Applications", *Knowledge Engineering Review,* vol. 11, pp. 93-155, 1996.

[150] O. Hartig, C. Bizer, and J.-C. Freytag, "Executing SPARQL Queries over the Web of Linked Data", in *The Semantic Web - ISWC 2009*, 2009, pp. 293-309. [doi:10.1007/978-3-642-04930-9_19]

[151] P. Avillach, F. Mougin, *et al.*, "A semantic approach for the homogeneous identification of events in eight patient databases: a contribution to the European eu-ADR project", *Medical Informatics,* vol. 160, pp. 1085-1089, 2009.

[152] P. Avillach, M. Joubert, *et al.*, "Design and evaluation of a semantic approach for the homogeneous identification of events in eight patient databases: a contribution to the European EU-ADR project", *Studies In Health Technology And Informatics,* vol. 160, pp. 1085-9, 2010.

[153] M. Stahl, I. R. Edwards, *et al.*, "Assessing the Impact of Drug Safety Signals from the WHO Database Presented inSIGNAL': Results from a Questionnaire of National Pharmacovigilance Centres", *Drug safety,* vol. 26, pp. 721-727, 2003.

[154] R. H. B. Meyboom, M. Lindquist, *et al.*, "Signal Selection and Follow-Up in Pharmacovigilance", *Drug safety,* vol. 25, pp. 459-465, 2002.

[155] M. Wadman, "Experts call for active surveillance of drug safety", *Nature,* vol. 446, pp. 358-359, 2007. [doi:10.1038/446358b]

[156] A. Bauer-Mehren, E. M. van Mullingen, *et al.*, "Automatic Filtering and Substantiation of Drug Safety Signals", *PLoS Comput Biology,* vol. 8, p. e1002457, 2012. [doi:10.1371/journal.pcbi.1002457]

[157] L. Härmark and A. C. Van Grootheest, "Pharmacovigilance: methods, recent developments and future perspectives", *European Journal of Clinical Pharmacology,* vol. 64, pp. 743-752, 2008.

[158] L. Wood and C. Martinez, "The General Practice Research Database: Role in Pharmacovigilance", *Drug safety,* vol. 27, pp. 871-881, 2004.

[159] J. Parkinson, S. Davis, and T. v. Staa, "The General Practice Research Database: Now and the Future", in *Pharmacovigilance*, ed: John Wiley & Sons, Ltd, 2006, pp. 341-348.

[160] P. Shannon, A. Markiel, *et al.*, "Cytoscape: a software environment for integrated models of biomolecular interaction networks", *Genome Research,* vol. 13, pp. 2498 - 504, 2003.

[161] P. M. Coloma, G. Trifirò, *et al.*, "Electronic healthcare databases for active drug safety surveillance: is there enough leverage?", *Pharmacoepidemiology and Drug Safety,* vol. 21, pp. 611-621, June 2012 2012. [doi:10.1002/pds.3197]

[162] G. Trifirò, V. Patadia, *et al.*, "EU-ADR healthcare database network vs. spontaneous reporting system database: preliminary comparison of signal detection", *Studies In Health Technology And Informatics,* vol. 166, pp. 25-30, 2011.

[163] T. Oinn, M. Addis, *et al.*, "Taverna: a tool for the composition and enactment of bioinformatics workflows", *Bioinformatics,* vol. 20, pp. 3045 - 3054, 2004.

[164] L. A. Zadeh, "A simple view of the Dempster-Shafer theory of evidence and its implication for the rule of combination", *AI magazine,* vol. 7, p. 85, 1986.

[165] E. Heard, S. Tishkoff, *et al.*, "Ten years of genetics and genomics: what have we achieved and where are we heading?", *Nat Rev Genet,* vol. 11, pp. 723-733, 2010.

[166] G. S. Ginsburg and J. J. McCarthy, "Personalized medicine: revolutionizing drug discovery and patient care", *Trends in Biotechnology,* vol. 19, pp. 491-496, 2001.

[167] A. D. Weston and L. Hood, "Systems Biology, Proteomics, and the Future of Health Care: Toward Predictive, Preventative, and Personalized Medicine", *Journal of Proteome Research,* vol. 3, pp. 179-196, 2004. [doi:10.1021/pr0499693]

[168] D. N. Cooper, J. M. Chen, *et al.*, "Genes, mutations, and human inherited disease at the dawn of the age of personalized genomics", *Human Mutation,* vol. 31, pp. 631-655, 2010. [doi:10.1002/humu.21260]

[169] T. I. H. Consortium, "A second generation human haplotype map of over 3.1 million SNPs", K. A. Frazer, D. G. Ballinger, *et al.*, Eds., ed: Nature Publishing Group, 2007.

[170] G. A. Thorisson, J. Muilu, and A. J. Brookes, "Genotype-phenotype databases: challenges and solutions for the post-genomic era", *Nat Rev Genet,* vol. 10, pp. 9-18, 2009.

[171] R. D. Hawkins, G. C. Hon, and B. Ren, "Next-generation genomics: an integrative approach", *Nat Rev Genet,* vol. 11, pp. 476-486, 2010.

[172] M. Muers, "Human genomics: Filling gaps and finding variants", *Nat Rev Genet,* vol. 11, p. 387, Jun 2010. [doi:10.1038/nrg2800]

[173] C. Mitropoulou, A. J. Webb, *et al.*, "Locus-specific database domain and data content analysis: evolution and content maturation toward clinical usea", *Human Mutation,* vol. 31, pp. 1109-1116, 2010. [doi:10.1002/humu.21332]

[174] J. T. den Dunnen and M. H. Paalman, "Standardizing mutation nomenclature: why bother?", *Human Mutation,* vol. 22, pp. 181-2, Sep 2003. [doi:10.1002/humu.10262]

[175] M. Wildeman, E. van Ophuizen, *et al.*, "Improving sequence variant descriptions in mutation databases and literature using the Mutalyzer sequence variation nomenclature checker", *Human Mutation,* vol. 29, pp. 6-13, Jan 2008. [doi:10.1002/humu.20654]

[176] R. Dalgleish, P. Flicek, *et al.*, "Locus Reference Genomic sequences: an improved basis for describing human DNA variants", *Genome Medicine,* vol. 2, p. 24, 2010.

[177] P. Riikonen and M. Vihinen, "MUTbase: maintenance and analysis of distributed mutation databases", *Bioinformatics,* vol. 15, pp. 852-859, October 1, 1999 1999. [doi:10.1093/bioinformatics/15.10.852]

[178] Christophe Béroud, Gwenaîlle Collod-Béroud, *et al.*, "UMD (Universal Mutation Database): A generic software to build and analyze locus-specific databases", *Human Mutation,* vol. 15, pp. 86-94, 2000.

[179] Ivo F.A.C. Fokkema, Johan T. den Dunnen, and Peter E.M. Taschner, "LOVD: Easy creation of a locus-specific sequence variation database using an ldquoLSDB-in-a-boxrdquo approach", *Human Mutation,* vol. 26, pp. 63-68, 2005.

[180] I. F. A. C. Fokkema, P. E. M. Taschner, *et al.*, "LOVD v.2.0: the next generation in gene variant databases", *Human Mutation,* vol. 32, pp. 557-563, 2011.

[181] J. T. den Dunnen, R. H. Sijmons, *et al.*, "Sharing data between LSDBs and central repositories", *Human Mutation,* vol. 30, pp. 493-495, 2009.

[182] C. Wu, C. Orozco, *et al.*, "BioGPS: an extensible and customizable portal for querying and organizing gene annotation resources", *Genome Biology,* vol. 10, p. R130, 2009. [doi:citeulike-article-id:6131194]

[183] Z. Li, X. Liu, *et al.*, "DRUMS: A human disease related unique gene mutation search engine", *Human Mutation,* vol. 32, pp. E2259-E2265, 2011.

[184] M. G. Aspinall and R. G. Hamermesh, "Realizing the promise of personalized medicine", *Harv Bus Rev,* vol. 85, pp. 108-17, 165, Oct 2007.

[185] W. Kalow, "Pharmacogenetics and personalised medicine", *Fundamental & Clinical Pharmacology,* vol. 16, pp. 337-342, 2002.

[186] J. N. Hirschhorn and M. J. Daly, "Genome-wide association studies for common diseases and complex traits", *Nat Rev Genet,* vol. 6, pp. 95-108, Feb 2005. [doi:10.1038/nrg1521]

[187] P. Lopes, D. Pinto, *et al.*, "Arabella - A Directed Web Crawler", in *International Conference on Knowledge Discovery and Information Retrieval*, Funchal - Madeira, Portugal, 2009, pp. 270-273.

[188] S. Matos, J. Arrais, *et al.*, "Concept-based query expansion for retrieving gene related publications from MEDLINE", *BMC Bioinformatics,* vol. 11, p. 212, 2010.

[189] A. J. Webb, G. A. Thorisson, and A. J. Brookes, "An informatics project and online "Knowledge Centre" supporting modern genotype-to-phenotype research", *Human Mutation,* vol. 32, pp. 543-550, May 2011. [doi:10.1002/humu.21469]

[190] N. Cannata, M. Schroder, *et al.*, "A Semantic Web for bioinformatics: goals, tools, systems, applications", *BMC Bioinformatics,* vol. 9, p. S1, 2008. [doi:10.1186/1471-2105-9-s4-s1]

[191] M. Swertz, E. de Brock, *et al.*, "Molecular Genetics Information System (MOLGENIS): alternatives in developing local experimental genomics databases", *Bioinformatics,* vol. 20, pp. 2075 - 2083, 2004.

[192] M. A. Swertz, K. J. Velde, *et al.*, "XGAP: a uniform and extensible data model and software platform for genotype and phenotype experiments", *Genome Biology,* vol. 11, p. R27, 2010. [doi:10.1186/gb-2010-11-3-r27]

[193] D. Kessner, M. Chambers, *et al.*, "ProteoWizard: open source software for rapid proteomics tools development", *Bioinformatics,* vol. 24, pp. 2534-2536, 2008.

[194] D. Glez-Peña, M. Reboiro-Jato, *et al.*, "AIBench: A rapid application development framework for translational research in biomedicine", *Computer Methods and Programs in Biomedicine,* vol. 98, pp. 191-203, 2010. [doi:10.1016/j.cmpb.2009.12.003]

[195] M. D. Wilkinson, B. Vandervalk, and L. McCarthy, "SADI Semantic Web Services - 'cause you can't always GET what you want!", in *IEEE Asia-Pacific Services Computing Conference*, 2009, pp. 13-18.

[196] M. Wilkinson, B. Vandervalk, and L. McCarthy, "The Semantic Automated Discovery and Integration (SADI) Web service Design-Pattern, API and Reference Implementation", *Journal of Biomedical Semantics,* vol. 2, p. 8, 2011.

[197] T. Slater, C. Bouton, and E. S. Huang, "Beyond data integration", *Drug Discovery Today,* vol. 13, pp. 584-589, 2008. [doi:10.1016/j.drudis.2008.01.008]

[198] S. Kozhenkov, Y. Dubinina, *et al.*, "BiologicalNetworks 2.0 - an integrative view of genome biology data", *BMC Bioinformatics,* vol. 11, p. 610, 2010.

[199] M. Hepp, "Semantic Web and semantic Web services: father and son or indivisible twins?", *Internet Computing, IEEE,* vol. 10, pp. 85-88, 2006.

[200] N. Cannata, E. Merelli, and R. B. Altman, "Time to Organize the Bioinformatics Resourceome", *PLoS Comput Biology,* vol. 1, p. e76, 2005.

[201] V. Maojo and F. Martin Sanchez, "Bioinformatics: Towards New Directions for Public Health", *Methods Inf Med,* vol. 43, pp. 208-214, 2004.

[202] J. Arrais, P. Lopes, and J. Oliveira, "Challenges Storing and Representing Biomedical Data", *Information Quality in e-Health,* pp. 53-62, 2011. [doi:10.1007/978-3-642-25364-5_6]

[203] J. H. Moore, F. W. Asselbergs, and S. M. Williams, "Bioinformatics challenges for genome-wide association studies", *Bioinformatics,* vol. 26, pp. 445-455, February 15, 2010 2010. [doi:10.1093/bioinformatics/btp713]

[204] G. H. Fernald, E. Capriotti, *et al.*, "Bioinformatics challenges for personalized medicine", *Bioinformatics,* vol. 27, pp. 1741-1748, July 1, 2011 2011. [doi:10.1093/bioinformatics/btr295]

[205] V. Marx, "My data are your data", *Nat Biotech,* vol. 30, pp. 509-511, 2012. [doi:10.1038/nbt.2243]

[206] Peter Haase, *et al.*, "An evaluation of approaches to federated query processing over linked data" in

[207] K.-H. Cheung, H. R. Frost, *et al.*, "A journey to Semantic Web query federation in the life sciences", in *6th International Conference on Semantic Systems*, Graz, Austria, 2010*BMC Bioinformatics,* vol. 10, 2009. [doi:10.1145/1839707.1839713]

[208] S. S. Sahoo, W. Halb, *et al.*, "A Survey of Current Approaches for Mapping of Relational Databases to RDF", *W3C,* 2009.

[209] G. Bumans and K. Cerans, "RDB2OWL: a practical approach for transforming RDB data into RDF/OWL", in *6th International Conference on Semantic Systems*, Graz, Austria, 2010, pp. 1-3. [doi:10.1145/1839707.1839739]

[210] M. Hazber, J. Yang, and Q. Jin, "Towards Integration Rules of Mapping from Relational Databases to Semantic Web Ontology", in *2010 International Conference on Web Information Systems and Mining*, 2010, pp. 335-339. [doi:10.1109/wism.2010.21]

[211] S. o. Auer, S. Dietzold, *et al.*, "Triplify: light-weight linked data publication from relational databases", in *18th international conference on World Wide Web*, Madrid, Spain, 2009, pp. 621-630. [doi:10.1145/1526709.1526793]

[212] C. Bizer and R. Cyganiak, "D2R Server - Publishing Relational Databases on the Semantic Web", 2006.

[213] J. Almeida, H. Deus, and W. Maass, "S3DB core: a framework for RDF generation and management in bioinformatics infrastructures", *BMC Bioinformatics,* vol. 11, p. 387, 2010.

[214] A. Shaban-Nejad, C. Baker, *et al.*, "The FungalWeb Ontology: Semantic Web Challenges in Bioinformatics and Genomics", in *The Semantic Web - ISWC 2005*, Galway, Ireland, 2005, pp. 1063-1066. [doi:10.1007/11574620_78]

[215] H. Nabarette, D. Oziel, *et al.*, "Use of a directory of specialized services and guidance in the healthcare system: the example of the Orphanet database for rare diseases", *Revue d'épidémiologie et de santé publique,* vol. 54, p. 41, 2006.

[216] E. Seoane-Vazquez, R. Rodriguez-Monguio, *et al.*, "Orphanet Journal of Rare Diseases", *Orphanet journal of rare diseases,* vol. 3, p. 33, 2008.

[217] A. Schieppati, J. I. Henter, *et al.*, "Why rare diseases are an important medical and social issue", *The Lancet,* vol. 371, pp. 2039-2041, 2008.

[218] S. Weibel, "The Dublin Core: A Simple Content Description Model for Electronic Resources", *Bulletin of the American Society for Information Science and Technology,* vol. 24, pp. 9-11, 1997. [doi:10.1002/bult.70]