universität wien

DIPLOMARBEIT

„A Distributed Data Aggregation Platform for Plant Metabolomics and Biochemistry“

Deutsch: “Eine distributierte Daten Aggregations Platform für Pflanzen Metabolomik und Biochemie”

Author

Lorenz Sauer, B.Sc.

angestrebter akademischer Grad

Magister der Naturwissenschaften (Magister rerum naturalium)

Wien, 14.05.2012

Studienkennzahl lt. Studienblatt:	A 490
Studienrichtung lt. Studienblatt:	Diplomstudium Molekulare Biologie
Betreuerin / Betreuer:	Univ.-Prof. Dr. Wolfram Weckwerth

Contents

List of Abbreviations

Acronym	Definition
AI	Artificial Intelligence
API	Application Programming Interface
CML	Chemistry Markup Language
CPU	central processing unit
EBI	European Bioinformatics Institute
EMBL	European Molecular Biology Laboratory
EMBL-Bank	European Molecular Biology Laboratory Nucleotide
Env	Environment
GUI	graphical user interface
JSON	JavaScript Object Notation
KEGG	Kyoto Encyclopedia of Genes and Genomes
LC-MS	Liquid-Chromatography Mass-Spectrometry
MS	mass spectrometry
MS Mass	Spectrometry
MS/MS	tandem mass spectrometry
MSn	n-tandem Mass Spectrometry
NCIB	National Cancer Institute Center for Bioinformatics
NIH	National Institutes of Health
PubMed	Free National Library of Medicine online database of
R	precursor to the letter S in the alphabet: Statistics package, inspired by S
RDF	Resource Description Framework
RNA	Ribonucleic Acid
RSS	Really Simple Syndication
SOA	service-oriented architecture
TCP/IP	Transmission Control Protocol/Internet Protocol
UML	Unified Modeling Language
UniProt	Universal Protein Resource
URI	Uniform Resource Identifier
VPN	Virtual Private Network
XML	Extensible Markup Language

Abstract

Metabolomics is a scientific discipline concerned with the unbiased measurement and characterization of all small molecular compounds within an organism. By dealing with metabolites, physical granularity is finer than in other *Omics* disciplines and, inextricably, the amount of data generation. The situation is aggravated by the exhaustive number of metabolites in plants, owing to a complex secondary plant metabolism. Metabolomics is tightly interwoven with Systems Biology, thereby gaining functional insight into plant processes. This knowledge is applied for instance in the quest of finding new remedies against diabetes, obesity and cancer. The Systems Biology and Metabolomics communities offer vast amounts of data, tools and supplementary information of specialist research areas, which are incrementally expanded and updated at many different locations and in many different data formats. Desirable to the field of Metabolomics and Biochemistry would be a quick up-to-date research-overview with unimpeded data accessibility, and without the need for deeper initial investigation into the nature of the underlying data and its formats.

Described herein are the motivation, implementation, challenges and future outlook of a Metabolomics summarization portal, and its underlying open-ended service-oriented architecture (*SOA*), amenable to aspects of distributed or cloud computing. Metabolomics, as a computationally demanding and data-intensive discipline, subject to a broad range of data types and distributed data-sets, is aided by the prototypal implementation of an aggregation web portal.

The underlying toolkit provides rapid application development (*RAD*) of scientific web-applications and Metabolomics related web-services.

Keywords: Arabidopsis / Plant Metabolomics / Aggregation / Web Portal / SOA / Database / Metabolomics / Visualization / Web Application

Kurzfassung

Metabolomics ist eine wissenschaftliche Disziplin die sich mit der unverfälschten Messung und Charakterisierungs aller kleinen molekularen Verbindungen innerhalb eines Organismus beschäftigt. Durch den Umgang mit Metaboliten, ist die physische Granularität feiner als in anderen *Omics*-Disziplinen und damit die Menge der Daten-Generierung. Die Daten-Situation wird durch die erschöpfende Zahl von Metaboliten in Pflanzen aufgrund eines komplexen sekundären pflanzlichen Stoffwechsel verschärft. Metabolomics ist eng mit der Systembiologie verknüpft und ermöglicht Einblick in Pflanzen-Prozesse. Diese Erkenntnisse werden beispielsweise bei der Suche neuer Heilmittel gegen Diabetes, Fettleibigkeit und Krebs angewendet. Die Systembiologie und Metabolomics-Community bietet eine große Fülle von Daten, Werkzeugen und ergänzende Informationen von spezialisierten Forschungsgebieten, die schrittweise erweitert und aktualisiert werden, - an unterschiedlichen Standorten und in unterschiedlichen Datenformaten. Wünschenswert, im Bereich der Metabolomik und Biochemie wäre eine schnelle, aktuelle Forschungs-Übersicht mit ungehinderten Zugriff auf Daten, und ohne die Notwendigkeit zur Untersuchung der Art der zugrunde liegenden Daten und deren Formate.

Beschrieben wird hierin, die Motivation, Implementierung, Herausforderung und Perspektiven eines Metabolomics Aggregation - Portals, und dessen zugrunde liegende, offene, service-orientierte Architektur (*SOA*),die gefügig für Aspekte des verteilten Rechnens ist. Metabolomics, als rechnerisch anspruchsvolle und datenintensive Disziplin mit einem vielen Datentypen und verstreuten Datensätzen, wird geholfen durch die prototypische Umsetzung eines Aggregations web-Portals.

Das zugrundeliegendes Toolkit ermöglicht die schnelle Entwicklung von wissenschaftlichen Web-Applikationen (*RAD*) und Metabolomics bezogenen Web-Services.

Keywords: Arabidopsis / Plant Metabolomics / Aggregation / Web Portal / SOA / Database / Metabolomics / Visualization / Web Application

Pathway analysis and Bioinformatics
Metabolomics
Proteomics
RNA Profiling
Genomics
(Meta) Data Mining and Integration

Preface and Acknowledgements

The thesis was created from May 2011 to April 2012, in part, at the Molecular Systems Biology (*MoSys*) group, located at the *University of Vienna*.

Dr. Wolfram Weckwerth (*MoSys* head of department and co-chair of the Multinational Arabidopsis Committee - MASC[1]) and members from the Metabolomics community are initiating the MASC Metabolomics forum, *MASCM*. This thesis could eventually serve the Metabolomics community, through the implementation of an aggregator platform, which pools concurrent metabolic data resources from the Web for uniform presentation within a common web-application interface.

Acknowledgements

I am thankful for my thesis supervisor *Univ. Prof. Dr. Wolfram Weckwerth* and particularly my thesis instructor *Dr. Volker Egelhofer* and all people at *MoSys* for the nice atmosphere and useful hints. I want to acknowledge the help from M.sc. *David Lyon*, M.sc. *Lena Fragner*, Dr. *Till Ischebbeck, Xiaoliang Sun Ph.D., Dr. Alexander Seidel* from MoSys[2] and fruitful discussions with Univ. Pro.f Dr. *Christoph Flamm* (TBI[3]) as well as *Peifen Zhang*, Ph.D. (PMN[4]), *João Ferreira*, Ph.D. (XLDB[5]), *Joshua Heazlewood,* Ph.D. (JBEI[6]), *James Treworgy* and others not explicitly mentioned.

The thesis is dedicated to my family, *Viktor* and *Christine* who supported me throughout my studies in ways only a parent is likely to understand, as well as my two brothers *Sebastian* and *Benedikt*.

[1] http://www.arabidopsis.org/portals/index.jsp
[2] www.univie.ac.at/mosys
[3] TBI Theoretical Biochemistry Group http://www.tbi.univie.ac.at/
[4] http://www.plantcyc.org/about/pmn_staff.faces
[5] http://xldb.fc.ul.pt/
[6] http://pbd.lbl.gov/

1. Introduction

1.1 Motivation

> ***"Genomics and proteomics tell you what might happen, but metabolomics tells you what actually did happen".***
>
> **- Bill Lasley, University of California, Davis**

Metabolomics is the unbiased identification and quantification of all metabolites in a biological system and is generally performed via Mass Spectrometry (MS) or NMR Spectroscopy. Metabolites are small molecular compounds of a molecular weight of usually less than 600 Dalton, spanning a wide range of chemical classes as well as concentration ranges of several orders of magnitude (Weckwerth W. , 2003).
Omics is a generic term, characterizing a wide range of science and engineering fields which generate vast amounts of data, in the quest of delineating the totality of functions and interactions that govern biological systems at various hierarchical levels. Increases in data acquisition are commonly driven by advancements in automation and measurement-technology, amenable to high-throughput methodologies. In other cases, Omics is driven by advances in computing and algorithms alone (Rihn, 2003). The obtained raw Metabolomics data-sets can be seen as timeseries-snapshots of equilibrated states of organisms, constituting a corpus of discrete *Omes* (Sakata, 2009). The number of metabolites in plants is much higher than in animals, owing to a complex secondary metabolism, estimated to exceed 200 000 metabolites (Weckwerth W. , 2003).

Ultimately, careful protocol consideration, data processing, statistical treatment and literature research puts focus on one or a set of metabolites as a potential candidate for *biomarkers*, as a participant in a *biochemical process*, for *verification of a hypothesis* through *systematic reviews* and *meta-analysis* as well as other research aspects. This thesis is concerned with the aggregation of literature, tools and data for- or around a particular metabolite found in the plant model organism *Arabidopsis thaliana*. Aggregation refers to the process of creating a compound object from several smaller ones (J Heaton, 2002). Data and literature research is generally a time consuming task, hindered by numerous publisher license/access schemas and a wide range of specialist research areas. This situation is improved by text mining methods which add

considerably to the extraction of systematic knowledge from the literature (Chikashi Nobata, 2011), and are often a key aspect of many biological and chemical databases hosted on the Web.

Due to the disorganized and semantically un-annotated nature of the Web, the search for suitable data-resources generally turns into a daunting and time-consuming task (HJ Joshi, 2011). Moreover, annual data growth follows an exponential law (Gavish, 2010)

A key feature of the Arabidopsis and other model plant communities is the abundance of online resources offering functional information. Often, studies employing mass-spectrometry lead to the creation of online resources for detailed investigation and interaction with the data (Weckwerth W, 2008). Locating and navigating these data resources can be difficult without intimate knowledge of the details that underlie the project which created and provisioned the data.

Usually researchers in plant Metabolomics seek out data resources on the Web through MS or NRM *spectra* data, *compounds*, *gene identifiers* (*ATG*'s in case of Arabidopsis thaliana (TAIR, 2000)), *Protein Accession* numbers, *Reactions*, *Pathways* or *Keywords* – thus constituting seven *search-types*.

Serving the researcher with unimpeded data-accessibility is a key goal of any summarization portal. *Pubmed* is a well known web-portal, initially conceived for searching the vast medical literature corpus, through primarily natural-language keyword searches. The query is underpinned by Pubmed-controlled synonymous vocabularies, or *MESH terms,* for the purpose of maximizing the reach of a particular keyword and thus the reach of the search and hence the number of qualitative results. For a proteomics portal, such as MASCP, access through a Protein name, Protein identifier or the Identifier of the underlying gene coding for a protein, is likely to cover most demands of the Proteomics community. In the field of Metabolomics however, the non-applicability of sequence data and direct application of the central dogma of molecular biology (F Crick, 1970), posit unique challenges in regard to presentation and cross-referencing of data. At the moment the field of Metabolomics is lacking a comprehensive, open and modular Data aggregation portal. Aggregators that currently may come closest to the project's objectives would be the proprietary *Chemspider*-Portal by RSC-Publishing and the open *Pubchem*-project by NCBI.

1.2 Goals

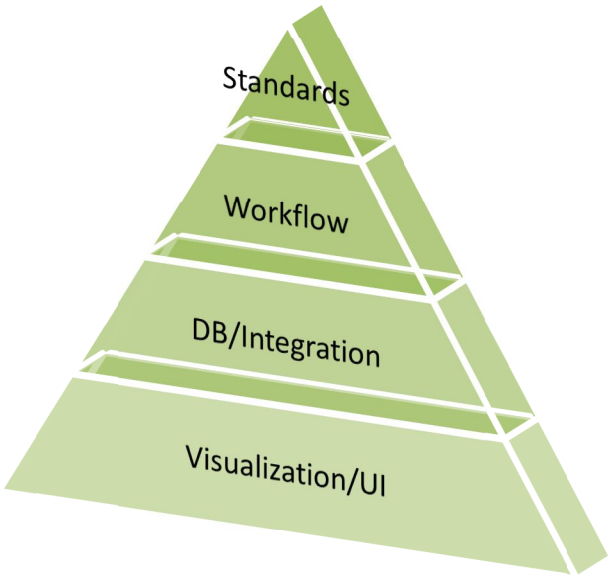


Fig 1 Visualizing the project's goals/aims and necessary effort

A Metabolomics web portal should present a logical workflow and uniform, unobtrusive user-interface, facilitate portal entry through the most frequent search-types which include Keywords, Spectra, Chemical Compounds and Gene Identifiers, and adapt to the information demands of the researcher. For instance, the initial search for new predictive biomarker candidates, with the premise of discovering indicators of a process, event, or condition (Schmidt, 2006), is less likely to require detailed physical, chemical and spectra information of a compound in contrast to literature, pathway, reaction and genetic information.

The main aspect of an aggregation portal is, firstly, the summarization of a particular set of web-resources which are queried on a subject in question, such as a Metabolite, and secondly, the presentation of the data, based on coherent user interface design aspects. The prerequisite for querying data is an implementation of inter-type and inter-accession conversion functions, which can translate between any of the given search types such as from a *gene identifier* to a set of *metabolites*, which occur in a reaction or, in another example, from *an enzyme classification number* to a *pathway*. Additionally, the source to be queried has to be well defined in terms of the input/output data formats and the web service-*endpoint*, which may be a simple url.

Further significant aims, whose approach is briefly discussed in the next paragraph, are the selection of an appropriate web-toolkit for a Browser plugin-free, uniform user interface, identification of usage-scenarios for the Aggregator, determination of data standards, selection of data-sources, creation of a federated schema and database, selection of data integration options, and concepts for data visualization.

1.3 Approach

To approach the implementation of a Plant Metabolomics Aggregator serving the Arabidopsis community, related Chemistry and MASC (Multinational Arabidopsis Steering Committee) projects were selected for investigation. To this end, the concept of the

two related aggregators, *Chemspider* and *MASCP*[7] (HJ Joshi, 2011), an aggregator serving the Arabidopsis Proteomics community, were studied.

Chemspider is a portal that pools chemical data from many different data providers, along with self-provisioned data from curated datasets. These datasets are obtained either on the basis of openly accessible data interfaces or through special negations with the data providers. A key issue in data-retrieval across databases of different projects is the creation and maintenance of a cross-reference database which can resolve a chemical species, given by its trivial name or structure, to the accession Identifiers and thus data entries of various data providers.

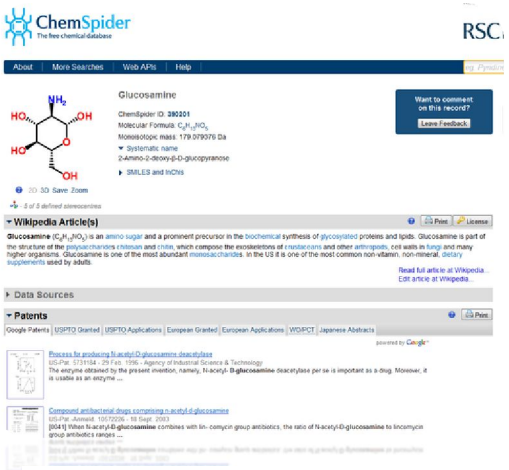


Fig 2 Chemspider as an example for a web-based aggregator for chemistry

Identified key-concepts of the MASCP project, which are also applicable to a Metabolomics portal, are the simple User interface, integrative data visualization and listings of Arabidopsis data-providers. Additionally the concept of using a vector graphics based user interface is well suited to the demands of scientific publishing.

Data Providers were chosen on the basis of related web-tools and Chemical or Arabidopsis related information (Akiyama K, 2008) (Garcia-Hernandez M, 2002) (HJ Joshi, 2011) (Weckwerth W, 2008). Included data providers encompass *KEGG* (Kanehisa M, 2000), Chebi (K Degtyarenko, 2007), *ProMex* (Hummel J, 2007), *TAIR* (Swarbreck D, 2008), *SUBA* (JL Heazlewood, 2007), RIKEN *PRIMe* (Akiyama K, 2008), *PlantMetabolomics.org* (Bais, 2010) , *Massbank* (H. Horai, 2010) and *Aracyc*/*PMN* (Mueller, 2003) (Zhang, 2005). The reasoning for inclusion, along a short project description and the provided data accessibility is stated in Chapter 2.

The Approach to **data integration** leaves several options, with Data Warehousing as one of the most intensive choices, yet offering unparalleled performance for Data Analysis. Alternatives are *Navigational Integration* and *Mediator-based Integration* (Hernandez T., 2004). Whereas in Navigational Integration the source/target combining occurs via a-hyperlinking model such as the Web, the latter option, Mediator-based Integration, couples data sources more closely, to be combined by a data wrapper, - the *Mediator*. Complex websites are often summarization portals themselves, and better amenable to be included via navigational integration in the form of *link-outs* or new

[7] http://www.masc-proteomics.org/mascp/index.php/Main_Page

document-views or '*tabs*' within the Aggregator's User Interface. Whenever data interfaces ('APIs') are directly available, Mediator-based methods are used, giving flexibility towards extensibility, data mixing, and presentation through the Mediator. Local data *federation* is realized for static data resources such as Enzyme Classification tables. The data integration process, and its various intermittent steps are shown schematically in Fig 3.

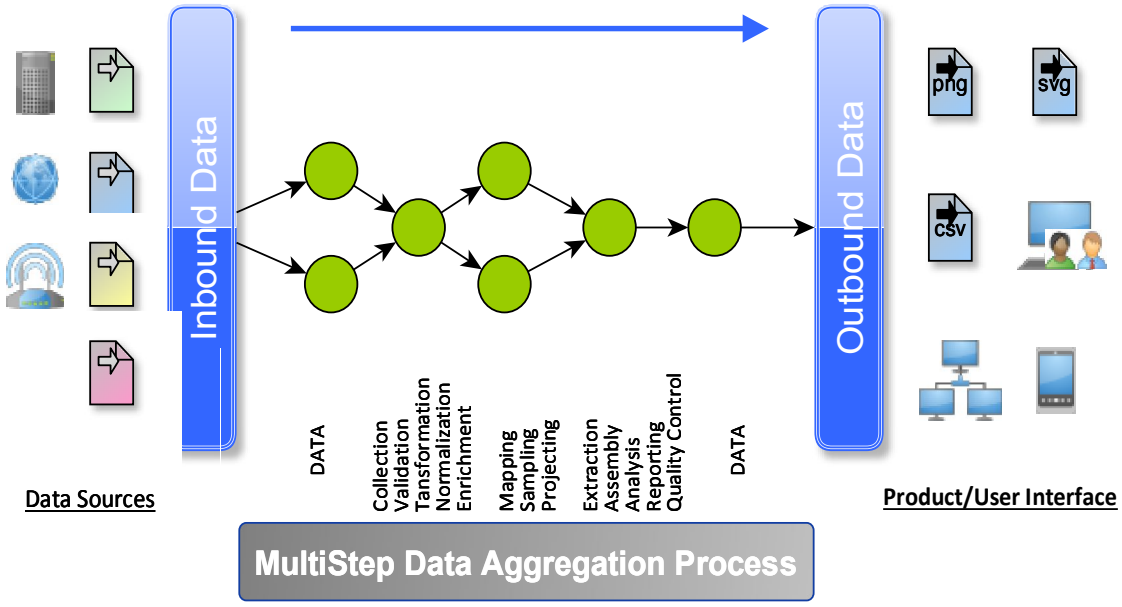


Fig 3 Showing the Data Aggregation Process and the numerous steps in aggregation that can occur intermittently. Data originates on the left, in different formats (represented by different colors), is collated, transformed and ultimately mapped to a visualization-interface or user interface element at the Client (right)

Querying the data not from a central source but from distributed projects with individual funding and individual data curators, offers several advantages. One is up-to-date data and distribution of responsibilities such as data management. Additionally the distributed data model shows less uncertainty overall, in terms of research funding and project support (HJ Joshi, 2011; D Reynolds, 2004).

The approach towards the **User Interface** aspects should gain initial attention during project planning, as it requires narrowing down a specific web-toolkit/library, often resulting in significant upfront learning-time investment. Professional large scale Bioinformatics Centers like EMBL and NCBI frequently employ the comprehensive ExtJS library (Sencha, 2011). Toolkit aspects to be taken into consideration are the licensing model, learning curve, documentation, performance and upfront time-investment, generally increasing with the capabilities of the library/toolkit. Comprehensive toolkits offer storage, communication, user-interface and visualization functions. For performance reasons and to meet demands of bioinformatic tools such as command para-

meter listings and manual-pages or '*manpages*', a purpose-written toolkit was incorporated, and connected to the lightweight user-interface library *jQueryUI*, which is introduced in Chapter 3.

The approach to **workflow** concepts incrementally improves during the development cycle with the addition of new functionality, revision of the existing functionality and abandoning or hiding user-interface features which are infrequently accessed. For instance, in an initial workflow scenario the search for a Reaction yielded reaction information via KEGG, Aracyc and TAIR as data providers. At a later stage the reactants were included in the compound visualization-matrix, and finally the workflow was extended to allow interactive browsing of reactions by clicking on each of the respective reactant to yield a list of related reactions. The reasoning behind these workflow decisions was tightly linked to the visualization and possible usage-scenarios, discussed next.

The approach towards appropriate **visualization** tools helps communication and interpretation of scientific data (W Cleveland, 1985).
In proteomics, dogma holds that the structure of a protein determines function and vice versa (J Skolnick, 2000). The structure is reflected in the protein's composition, which at its most basic level consists out of a linear chain of amino-acids, the primary structure. Mapping supplementary Omics data on top of a presentation of the protein's primary structure is thus a frequently used strategy (F Azuaje, 2005), and one that is employed in the MASCP Gator. Contrarily, Metabolomics data can be more faceted, and often requires pioneering consideration towards visualizations. Additionally, unlike complete sequence information, complete pathway information is hard to obtain and often incomplete (W Weckwerth, 2002). Whereas the correlation of gene and protein expression is low, it is even lower between gene expression and metabolites (U Roessner, 2009). Thus it is favorable to delineate enzymes and their corresponding genetic information against metabolic information in an integrative visualization context. A key visualization method in metabolomics involves overlaying metabolite concentrations with metabolic pathways (N. Gehlenborg, 2010).
One newly devised strategy for data visualization is through a linear assembly of clustered pathways or 'superpathways', arranged into a dendrogram to reveal the Pathway Semantic Similarity (Tiago Grego, 2010) between pathways. The vertical axis of the graphical 2D plot is constituted by a linear arrangement of metabolites, which too are amenable to hierarchical clustering through a dendrogram, based on similarity measures such as the Tanimoto Index (TT Tanimoto, 1957) which may be visually augmented by color-coding. Other distance measures for the horizontal pathway-dendrogram may be parameters based on enzymatic sequence homology, overlap in

EC reaction-types/species between pathways (Tohsato Y, 2000) or heuristic pathway similarity based on preceding studies (Jiang K, 2011).
Chemical species may be assigned to categories or hierarchies which could be retrieved from ontological databases rather than a linear arrangement of pathways. This is partially realized by coloring the pathway-clusters in the KEGG-metabolism color schema (Kanehisa Laboratories, 2012), wherein for instance *orange* represents pathways related to amino-acid metabolism. The same color coding concept could be extended and applied to metabolite-ontologies, or controlled vocabularies, provided by the *ChEBI Database* (K Degtyarenko, 2007). This would allow a functional presentation for many metabolites, but completeness of such information is lacking at the moment (McShan, 2005). Currently there is no 'metabolite ontology' database to systematically annotate individual compounds into different groups (J G Bundy, 2008).

1.4 Recapitulation

Plant Metabolomics deals with a large number of metabolites which are described in a large scientific literature and data corpus. The Metabolomics community is served by numerous scientific tools, which can be hard to find and data is spread over a large number of disparate databases, facing exponential growth. Arabidopsis thaliana as a popular plant model organism has an active community providing functional characterizations. To quickly get a comprehensive overview of metabolism-related research data and literature a unified aggregator portal is desirable. The aim of creating an aggregator lies in a uniform user-interface design, deciding which data sources to hyperlink, which to locally federate in a database, and which to query from the Web, as well as deciding upon the data standards to use, the workflow for the enduser, and creating visualizations of the integrated or ‘compounded’ data.

1.5 Thesis Organization

The structure of this work is as follows:
Chapter 2 provides a review of technical concepts and databases used for the implementation of the aggregator and its toolkit.
Chapter 3 briefly elaborates the issue of data collection, and describes the target audience of the Aggregator, and the implementation for the system design prototype
Chapter 4 introduces and analyzes important User Interface aspects.
Finally, *Chapter 5* concludes this work with a project-recapitulation, results and discussions, and a future outlook.

2. Basics and Methods

This chapter provides a review of technical concepts and databases used for the implementation of the aggregator and its toolkit.

The following sub-chapters can be divided into the topics *Server*, *Client*, *Presentation* and *Data* – in that order. Data is generally retrieved, distributed and stored at the *Server*, the *Web* and the *Client*, depending on the nature of the data. The *Client* connects to the *Server*, upon which the requested data is loaded from Web-accessible Databases, and conjoined with *Server*-side data for subsequent data collection, collation and enrichment, and finally data-aggregates are sent to the *Client* for the purpose of presentation (Fig 4).

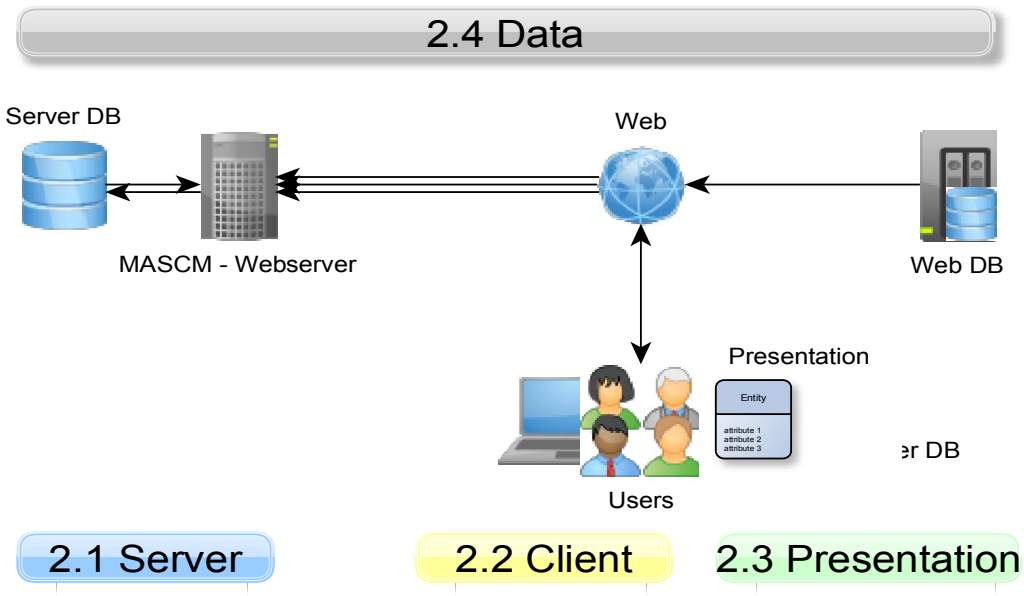


Fig 4 Showing an outline of the Server-Client relationship, and the distribution of the data. Numbers indicate the corresponding sub-chapter. *Web-DB* represents any generic data resource which is queried by the Server from the Web. Browser-DB is an HTML5-specified, small (>5MB) persistent database used for caching most frequently accessed data.
In this picture, the Web (blue globe) is assigned the role as a central network-hub.

2.1 Server

2.1.1 Distributed Computing

A computational grid is a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities (Foster, 1999). Distributed Computing refers to the federation of computer resources from disparate locations or domains for use under a common objective (B Fran, 2003). Distribution affords robustness and scalability, which is the property to handle growing demand on processing, storage or networking traffic in an enlarging manner, to accommodate that growth. By making all server aspects of the Aggregator portal distributable, flexibility is afforded at later stages of the portal's evolution, wherein computationally demanding services such as Spectra Deconvolution, automatic Spectra Annotation or quality control of large mass-spectral Metabolomics datasets could be offered. Secondly the distributed-model enforces a modular and extensible codebase, to interoperate seamlessly with functions or programs written by other developers around the world. In practice this means that disparate command-line programs, application-scripts, background-processes and external web-services can be federated and mapped to the Client-application.
This project facilitates the modular distribution of services through assembly of functions into service-domains assigned at specific service-endpoints, which can be hosted on a traditional web-server but also in *the Cloud* at a service Provider such as *Google*, *Amazon*, *Microsoft* or *Cloudera*. Since the same service-endpoints are hosted multiple times, in a redundant manner, an increase in availability and robustness is afforded.
The concepts works by a) furnishing the Client with a list of multiple endpoints for each service domain, b) initially randomly assigning service-endpoints to the connecting clients, and lastly, by c) letting the Client program switch an endpoint after a short timeout-period. The aim is to provide a good tradeoff between availability of the portal/web-services and access speed.
Distribution of the Server database can be performed via the Database Management System or DBMS, in this case MySQL, which can be set up to perform replication tasks within the server environment and between servers.
The distributed aspects are delineated in detail as part of *Chapter 3*.

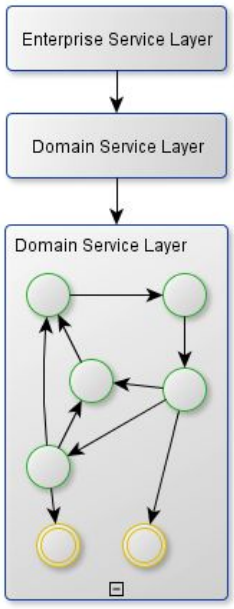


Fig 5 SOA layers in the Domain Model. The Domain service layer contains interoperable functions within separate domains.

2.1.2 Service Oriented Architecture (SOA) and APIs

Modularity of software components is the result of adherence to *SOA*-principles.
Service-Oriented Architecture (SOA) refers to a set of software design principles following the idea of developing discrete,

able software components (Fig 6), which can be repurposed at will. Many scientific applications serving the scientific communities are not web-service-oriented, despite a growing need to base applications on a service-oriented architecture (Fang L., 2006).

In SOA projects, disparate applications are connected on the basis of a common set of Interfaces, which are defined in terms of underlying languages and protocols, usually with the aim of cooperation within a web-based environment. Generally, all applications or modules share a loose dependency with their native operating system and OS dependent platforms. However, SOA separates service functionality into discrete units, accessible over the network through a defined data-interface (Fig 5)

An Application Programmable Interface (*API*) provides a data interface for software, which is ideally well-accessible in terms of hierarchy, data throughput and the learning curve for a potential API-user.

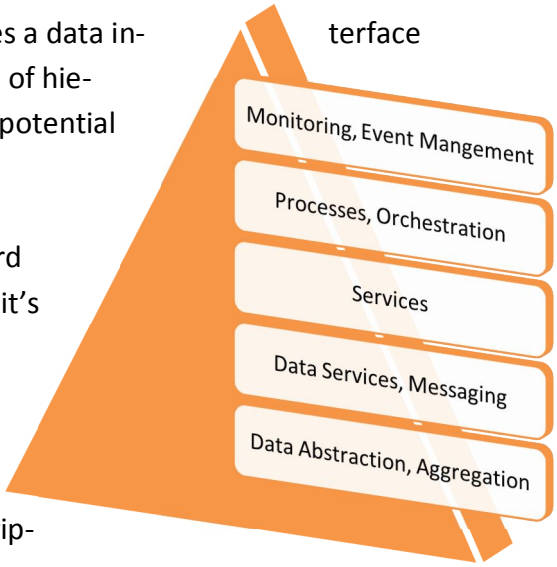


Fig 6 The SOA process hierarchy

Following is a list of web technologies which undergird the provision of the Aggregator web-application and it's APIs (Fig 5):

- **RDF**, **RSS** and similar standards which are part of the semantic web can be used to provide dynamic content updates on a subscription-basis across various interoperable applications
- **JSON** or Javascript Object Notation, provides a lightweight data-container for passing data between the server and client, when the capabilities of XML are not required (see 2.4.3)
- **SOAP** short for Simple Object Access Protocol provides Remote Procedure Calls (*RPC*, see 2.2.1) enveloped in XML over HTTP (see 3.2.1)
- **REST** or Representational State Transfer, builds upon the traditional HTTP protocol as a simple service-architecture based on url-paths instead of url-parameters. RESTful services are lightweight and broadly accessible (see 3.2.2)
- **AJAX** or Asynchronous JavaScript And XML, allows loading of data such as XML documents from a remote web resource and is typically used to enable site-updates without page-refreshes. AJAX is used to shuttle JSON messages between the server and client (See 2.4.3, 3.5.2.5).

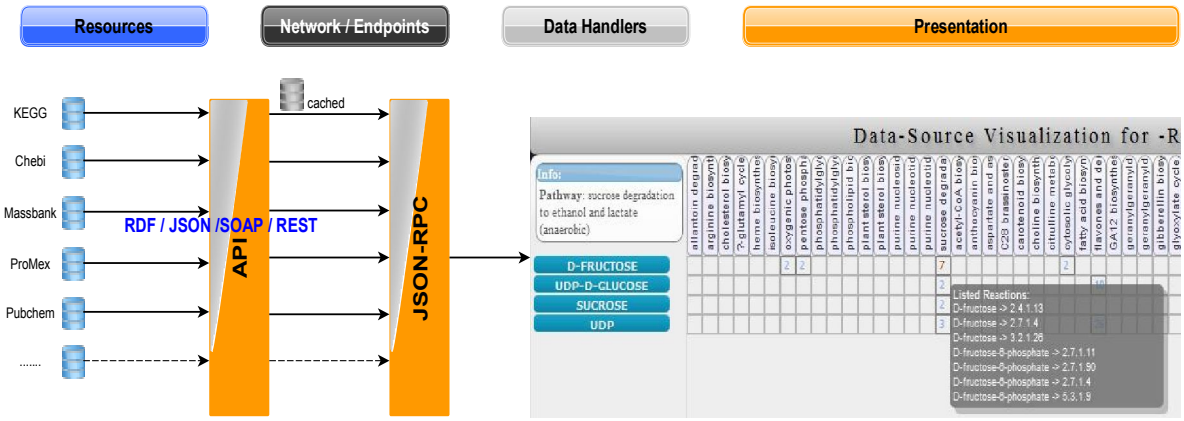


Fig 7 Schematic diagram of the data retrieval process for the Aggregator portal. Data is requested from online resources and cached, or from a locally federated Database (e.g. enzyme-classification information) through different API's. Different caching strategies are employed for different data, which is subsequently and disseminated via a JSON-RPC API. Event-based data handlers and data adaptors on the Client, incorporate the data incrementally into the presentation layer. Event-based strategies are generally used for asynchronous operations wherein the operation-sequence and schedule or even successful completion cannot be predicted. XHR2 refers to an AJAX specification.
Note: Distributed aspects are omitted for clarity.

2.2 Client

The Client or User accesses the Aggregator application via a HTML5 compatible web-browser. HTML5 is a W3C (World Wide Web Consortium) provisional standard, which aims to unify the programmability and demands of modern web-applications across browsers, without the need of third-party plugins. To facilitate communication between the Server and Client a JSON-RPC gateway is used (Fig 7).

2.2.1 RPC – Remote Procedure Call

Remote Procedure Calls or RPCs, allow programs to invoke each other's functions or procedures over a network, so long as these functions are public. RPCs are an old standing topic of networked software architecture. A *procedure,* in programmatic terms, refers to a function typically consisting of *inputs*, a *processing-declaration or function-routine* and *outputs*, all of which are optional, except for a *function declaration*. An exemplary function or procedure in the JavaScript browser-environment may be: `alert('hello world')`. Herein, the native procedure `alert` is passed as input a *character-sequence* or *string*, whereupon a popup dialog opens which displays the passed text (Fig 8) as part of the function's routine, without returning any output value. Technically *alert* is a wrapper-function for a native UI call of the underlying operating system which is mediated by the web-browser. On windows machines this will

result in *Win32API* function invocations.

Fig 8 Showing the resulting popup dialog of the native user-interface (more precisely *modal-dialog*), invoked by calling the JS-function *alert,* requiring the user to click on *OK* in order to continue using the Browser application user-interface

2.3 Presentation: User Interface Concepts

2.3.1 MVC - Model View Controller

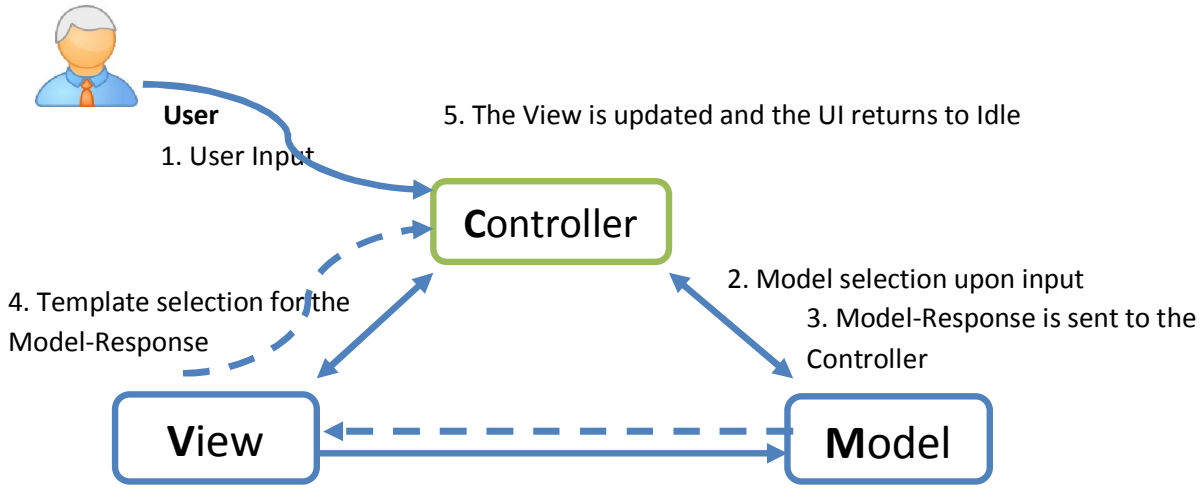


Fig 9 Model View Controller schematics. ___Solid line: direct association; …..Dashed line: indirect association e.g. stateless queries ('observers');
MVC allows different presentation layers without changing the underlying controller code

MVC describes the separation of the main aspects of an application comprised of *Input-*, *Business-* and *User Interface-* logic, with only loose coupling remaining between those elements. An easy way to understand MVC, is by assuming the *model* as the data, the *view* as the window on the user-screen, and the *controller* as the adhesive between the view and the model (Fig 9).
MVC as a programming paradigm which, when properly adhered to during the software-development-cycle of an application, allows for a flexible program layout of interoperating components. A practical example for an inter-operation is the search of a

spectrum, yielding three top compound candidates, and the subsequent retrieval of chemical information for these compounds from *PubChem*, to be finally displayed sparsely, in an interactive table with the option of table-data export to an open data-format.

A program adhering to the MVC concept can easily switch its graphical interface implementation, for instance from a *Windows Forms* application of the Dot Net Platform to a HTML-based user-interface rendered in a web-browser, incurring few to no changes in the underlying Controller logic.

MVC distinguishes between three aspects, comprising the *Data model* (Model), the *Presentation layer* (View) and the *Program logic* (Controller). One particularly important facet of MVC, is the reusability of flexible components to enable rapid implementation of various program-drafts, during the development-cycle of a particular application.

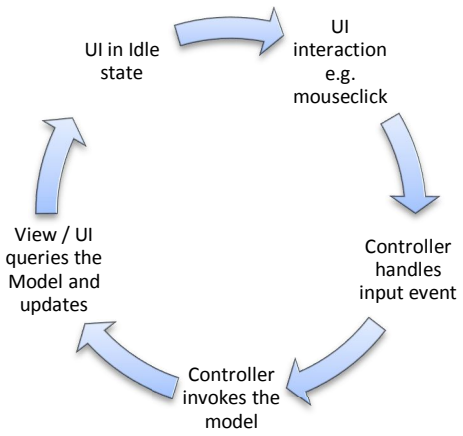


Fig 10 Showing the repeating process flow within an MVC application, and how MVC's three components (Model, View and Controller) are coupled to each other. Practically Model – View communication occurs asynchronously due to inherent networking latencies in the Web.

MVC process flow

As shown in Fig 10, the user interacts with the user interface in restricted ways, for instance by clicking on a button, as part of an element of the *View*.
The *Controller* handles the input event which is fired by the *user interface*, -usually via a registered handler or *callback function*. The callback function is invoked by the fired event, thereby converting the event into an appropriate user action which is understandable by the *Model*.
The *Controller* notifies the *model* of the user *action*, possibly resulting in a change of the model's state. For example, the controller might update a user's shopping cart.
Following, the *View* queries the *Model* in order to generate an appropriate re-rendering of the user interface. For example the *View* might list the user's shopping cart contents. The *View* can either retrieve its data from the *Model* or in other implementations, be instructed by the *Controller* to re-render itself appropriately, thus avoiding latencies incurred through *Model – View* communication. Lastly, the user interface waits idly for further user interactions, therewith restarting the control-flow cycle.

→ An MVC example is provided in 3.5.2.5.

2.3.2 The Web-Browser Document Object Model (DOM)

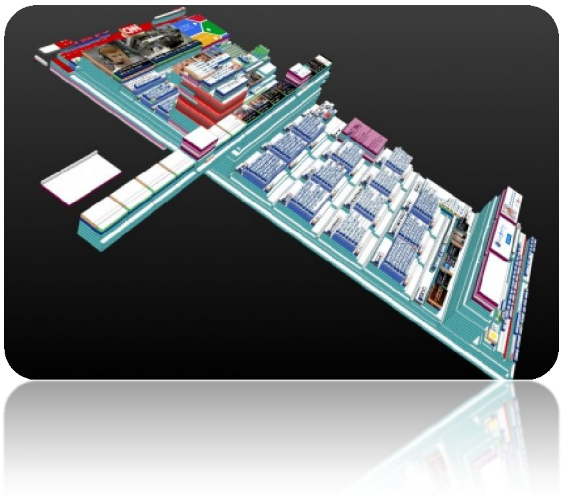


Fig 11 Showing the DOM structure as a height-map model of the *cnn.com* news website; The height corresponds to the number of nesting for a given DOM-element; The picture is rendered in WebGL[8]

```
▼Document
  <!DOCTYPE html>
  ▼<html>
    ▼<head>
       <title>html document title</title>
     </head>
     <body></body>
  </html>
```

Fig 5 Example of a simple html-document. Indentation reflects the level of nesting

The *DOM* is a *W3C* standardized programming model in which markup elements with various *rendering intents*, are assigned in a hierarchical tree-like order (Fig 11). Nested sub-elements inherit properties from elements of higher order such as visibility. Through the use of the same standardized API i.e. the same methods, DOM elements, which are part of a *document*, can be read or written via the syntax of a wide range of programming languages.

→ A DOM application example is provided in the Appendix.

DOM usage

The DOM is a ubiquitous and wide-spread object-model used throughout the project to treat web-documents in an object-oriented manner. On the server side the DOM is used for rule-based content-extraction of markup documents such as HTML documents. On the client-side, the DOM provides element selection abilities to enable updating of the user-interface and visual display of data.

[8] http://hacks.mozilla.org/2011/12/new-developer-tools-in-firefox-11-aurora/ Kevin Dangoor, December 23, 2011

Chapter 3 shows using the DOM in the context of this project's toolkit (See 3.5.2).

2.4 Data: Important File formats, Data structures and Databases

Similarly to the popular MIAME standard in the microarray field, for data exchange with metadata describing the experiment, the MIAMET (Minimum Information About a METabolomics experiment) standard is set to archive the same goal for Metabolomics data (O. Fiehn, 2007). Unfortunately adoption is still lacking, thus at the moment metadata is primarily shown by *linking out* to the original web- data source. Each metabolomics experiment should contain information about its design, samples, sample preparation, metabolite extraction and derivation, metabolic profiling design, metabolite measurement and measurement specifications. These aspects are heeded in a potential data-schema used for storing original experimental data, and is provided in the Appendix (See also Fig 15).

2.4.1 InChi9 & InChiKey and MDL mol-files

As a chemistry-centric aggregation project the project has to cope with the issue of converging on one and the same chemical species from a wide range of possible data access identifiers and compound formats. Some of these formats enrich or lose data during conversion. The IUPAC-adopted International Chemical Identifier (InChI) can uniquely describe chemical compounds, and should thus be constant preset in a chemical data-schema. The adoption of InChi-strings in databases is recommended to enable easy and unambiguous access across databases and web-services, - currently a daunting problem, given that many web-services require a unique accession Id for the retrieval of entries of a given database (InChi-Trust, 2011). (See 3.3) To resolve between entries of different databases, *cross-reference databases* and *cross-reference web-services* are required, which are difficult to maintain, usually not comprehensive and entail additional database queries, thus incurring additional latencies.
IUPAC provides a cross-platform conversion utility that reads several chemical input formats such as MDL's mol files and converts these to InChi representations (InChi, 2011). A Python script was devised to convert all KEGG compounds from mol-files into InChi entries via an IUPAC provided converter. InChiKeys must not be unique and are generated by applying a hashfunction on the InChi-string, yielding a much shorter representation. InChi strings can be converted into the popular SMILES string representation, which is used for on-the-fly generation of molecular representations through-

[9] http:// http://www.iupac.org/inchi/

out the web-portal (see 3.5.2.6). SMILES is plagued by different canonical string-output of different vendors.

Following, the *KEGG* Compound *C00074*[10] *Phosphoenolpyruvate (PEP)* is converted into an InChi representation via the IUPAC/InChi provided converter tool:

```
inchi-1 C00074.mol C00074.inchi
```

The resulting InChi file containing the InChi representation (3rd line) for PEP is then:

```
* Input_File: "~/Chemlib/keggcmpd_00074.mol"
Structure: 1
InChI=1S/C3H5O6P/c1-2(3(4)5)9-10(6,7)8/h1H2,(H,4,5)(H2,6,7,8)
```

Both SMILES and InChi data is stored in a chemical compound dataset on the server database, based on the list of KEGG compounds (see 2.5.1).

2.4.2 SDF files

The Structure Data Format or SDF is a wrapper of *molfile-data*, which specifies additional metadata or associated-molecular data to be placed at the end of the file. SDF files have a typical footer, consisting out of one line with four dollar signs: `$$$$` or in Regular Expression (*RegExp*) notation: `/\${4}$/`
Preceding the footer, are several line-wise entries, for instance:

```
>  <Unique_ID>
C00074
>  <ClogP>
6.12
```

Both, *sdf* and *mol files*, can be used in the compound search input, by generating corresponding SMILES or InChi strings enabling lookup of compounds in the server-database and subsequent invocation of aggregation-services.

2.4.3 JSON

JSON, short for *JavaScript Object Notation*, is a subset of the *JavaScript* language notation to serialize or store nested data-types in a lean, minimalistic *ASCII*-text based character sequence. JSON has found widespread application due to the ubiquitous use of JavaScript in web-browsers and as a lightweight data-container-alternative to XML.

[10] http://www.genome.jp/dbget-bin/www_bget?C00074 Phosphoenolpyruvate; PEP

Objects are encapsulated by curly brackets, and can be used as a *key-value store* separated by a colon, as follows '*key:value*'. Values can be any valid *JavaScript* data-type such as *numbers*, *booleans* (true|false), *arrays*, *strings* and other *objects*, thus allowing nesting. Cyclic references are not possible, effectively disallowing the passing of functions and function containing objects (both of which are marked by a cyclic prototype-chain in JavaScript, which is beyond the scope of this paragraph).
Following is a concise JSON data container example.

```
{"header":{"name":"Uridine 5'-diphospho-N-acetylglucosamine" ,"mass":73}
,"peaks":{"40":0,"41":13,"42":4,"43":39,"44":25,"45":207},"summary":["library
","10"],"author":null}
```

2.4.4 MSL, MSP – Spectra list files

MSP files are simple semicolon separated mass spectra text-files containing m/z peaks and corresponding intensities, such as for instance `40  0;  41  13;...`. Prior to the peak-listing, a header provides optional meta-information about the name, peak-purity, and the number of peaks in form of a *field-key* separated by a colon, followed by the *field-value* e.g. `Mass: 73`. Each data field starts at a *new line*.
MSL or *MS-Library* files are obtained through concatenation of multiple msp files into a single file, usually separated by a *new line*. MSL and MSP spectra are widely used for Electron Impact spectra.
A client-function implemented in JavaScript (*gator.spec.js*), allows dragging and dropping of msl-, or similar text-based spectra in the *spectra search tab*, by parsing the msp or msl-file. Thereupon peaklists are extracted into a data-structure, whilst omitting peaks with zero intensity and peaks below a user-defined threshold upon user-confirmation. A threshold value is suggested based on the peak just following the peak-window of the 33% most intense peaks, should more than ten peaks be present in the spectrum. The value is additionally rounded ('ceiled') to the next full decade, with intend of introducing less variance in the search parameters between spectra. (Fig 12). If the initial baseline cutoff prior to spectra import is set too high, the aforementioned routine does not apply.

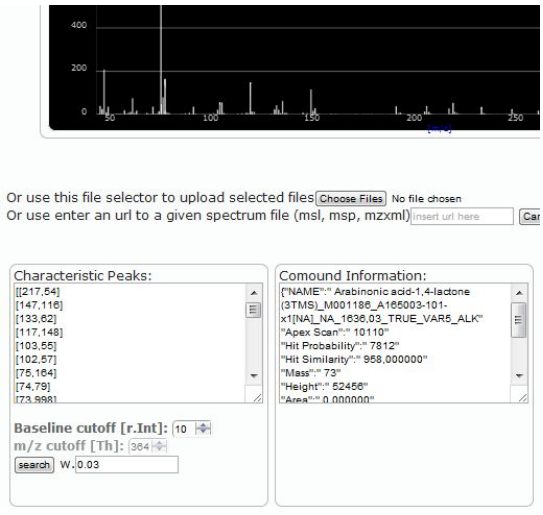


Fig 12 Showing the baseline cutoff estimated and ceiled at an intensity of 10 for an experimentally obtained sample spectrum, provided in the Appendix.

Spectra are stored on the server, and a database log entry is submitted. If the user is

logged in, this entry encompasses the User-ID, and the option to share spectra with other users becomes possible. Thus the web-portal may be used as a cloud-like storage platform.

Following is a short excerpt of an msp spectrum to demonstrate the format's outline.

```
NAME: Uridine 5'-diphospho-N-acetylglucosamine; GC-EI-TOF; MS; n TMS; BP_73
Mass: 73
Num Peaks: 561
  40   0;  41  13;  42   4;  43  39;  44  25;  45 207;  46  10;  47  36;  48   0;  49   0;
  50   0;  51   3;  52   2;  53   3;  54   1;  55  19;  56   4;  57   8;  58  12;  59  75;
```

→ The full msp-Electron-Impact (EI) spectrum, acquired on a GC/MS-TOF metabolomics experiment of Arabidopsis inflorescence measurements and exported via *Leco ChromaTOF*, is provided in the Appendix.

2.4.5 Databases and Data Extraction-Transform-Loading (ETL)

Databases

Database refers to the structured collection of data to furnish data-reliability and eased data access, of generally digital data assets (Ullman, 1983). Special demands or 'purposes' of the data organization determine the choice of database management system (*DBMS*).

Data is often modeled to address aspects of reality and is organized accordingly through data schemas (M Johnson, 2002).

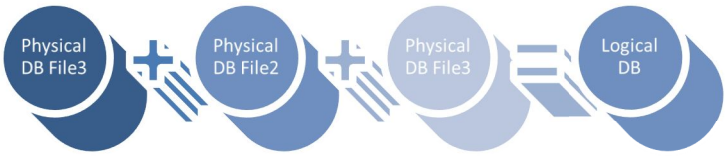


Fig 13 Logical vs. Physical Database

Initially databases were simple applications, with the purpose of storing data in digital form and granting quick access. This simple concept still applies in the field of biology through the widespread availability of structured text-files or *flatfiles*. Access speed necessitates a layered organization and hierarchy of the data which the database application has to understand. A *Logical Database* is an abstraction wherein the Database System manages the physical distribution and location of Data (MSDN, 2011)in accordance with the minimum requirements of Databases – the *ACID* rules of *Atomicity*, *Consistency*, *Isolation* and *Durability* (H Theo, 1983). Contrary to the former, a *Physical Database* is aware of the data-location of the logical

database and the operational paths between them (Fig 13). In common parlance, the term *Database* always refers to the *Logical Database* unless indicated otherwise. *MySQL*[11] is employed as the primary DBMS for the aggregator, owing to its widespread adoption, open source status, documentation, speed and extensibility. The Aggregator caches most data requests for a predefined time-period, which depends on the data-type, upon user request. Caching facilitates gains in access speed and availability of the provided web-services as well as for usage analysis to aid in carving out usage scenarios on which to focus future development (See 1.3, 5.2).

2.4.5.1 CRUD - Database Transactions

- `SELECT ... FROM ... WHERE ...`
- `INSERT INTO ... VALUES ...`
- `UPDATE ... SET ... WHERE ...`
- `DELETE FROM ... WHERE ...`

Fig 14 Typical CRUD (Create Read Update Delete) - Command Set of SQL (Note: text is web-linked)

Transaction-Oriented Databases are supposed to implement a set of rules that allow for a maximum of data consistency in situations of system failures, such as a computer-crash. Fault-tolerance is established by encapsulating database *CRUD operations* within transactions, which are said to be *atomic* if the rules of *Atomicity*, *Consistency*, *Isolation* and *Durability* (*ACID*) are obeyed. The primary objective for a Database during a system crash is not the prevention of any data-loss, but to ensure consistency (correct functioning). Database *operations* generally entail Database-*transactions*, yet operations do not necessarily result in a transaction. Both terms are often not sufficiently delineated and used interchangeably.

The four most basic operations of persistent-storage solutions are *Create*, *Read*, *Update* and *Delete*, collectively known as *CRUD-operations* (Fig 14) (Heller, 2007). Notably, CRUD operations are defined for the Hypertext Transfer Protocol (HTTP) and used in a stateless manner for *REST*-API interfaces also called *RESTful* when all CRUD operations are supported. Data Operations on Databases are performed via a Data Manipulation Language (*DML*), with the salient aspect of the *functional intend* being placed first (Fig 8). A particularly important DML is the Simple or Structured Query Language or *SQL* (EF Codd, 1970).

An important aspect for storing datasets in relational databases is initial delineating of data relations to yield reliable data-schemas and relations between them. Subsequently naïve data is imported from an arbitrary data source into the created database-tables which satisfy the relational schema.
Biological flatfile databases often contain redundant columns, rather than a data-relation to a separate table, as would be the case in a relational database. The step of

[11] www.mysql.com , MySQL © 2012, Oracle Corporation

extracting data, transforming and finally loading of the data into a DBMS is referred to as *ETL* (See 2.4.5.2).

Relational tables should show little redundancy in the assignment of keys and values and the relations between them, referred to as *normalized*, which is exemplified in Fig 9 for the TAIR flatfile database *AraCyc_Enzymes_Without_Tags_BLASTset.fasta*[12]. The flatfile was created from a FASTA Protein sequence file via regular expressions and a custom-written Python script.

```
DROP TABLE IF EXISTS `kegg`.`arath_tair_enzymesequence`;
CREATE TABLE  `kegg`.`arath_tair_enzymesequence` (
  `uniprot_id` varchar(25) PRIMARY KEY NOT NULL,
  `gene_id` varchar(25) NOT NULL,
  `unit` varchar(60) NOT NULL,
  `description` mediumtext,
  `species` varchar(100) NOT NULL,
  `genes` varchar(100) NOT NULL,
  `sequence` mediumtext NOT NULL,
  KEY `i_all` (`uniprot_id`,`gene_id`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
CREATE INDEX `i_all` USING BTREE ON
  `kegg`.`arath_tair_enzymesequence` (`uniprot_id`,`gene_id`);
ALTER TABLE `kegg`.`arath_tair_enzymesequence` DROP INDEX `i_all`;
```

Fig15 - Exemplary Database Table; Four separate SQL statements are shown, terminated by a semicolon. The first statement deletes the table without warning should it already exist. The CREATE TABLE operation is followed by a table.name and the declaration of the table data-schema. An INDEX 'i_all' is defined for uniprot_id and gene_id, and the storage engine is set to MyISAM (MySQL specific implementation of the *Indexed Sequential Access Method*).For automated-relations the Engine should be set to the newer *InnoDB* format. The characterset is set to be Unicode compatible.
The third SQL statement demonstrates how to independently create Indices in an existing table; likewise '*DROP*' can be used to delete Indices (fourth SQL statement).
The relations of the table are *uniprot_id*, *gene_id* – which link to other tables. '*species*' could be indexed and made into a further relation, but at the moment the species is limited to Arabidopsis thaliana.

2.4.5.2 ETL Processes

Extract, *Transform*, *Load* (ETL) refers to the chronology of the process by which data from usually different, heterogeneous sources is integrated into a target database. ETL-tasks (Fig 16) are ascribed a paramount role for the establishment of a federated Database. As entire databases are integrated the term data-

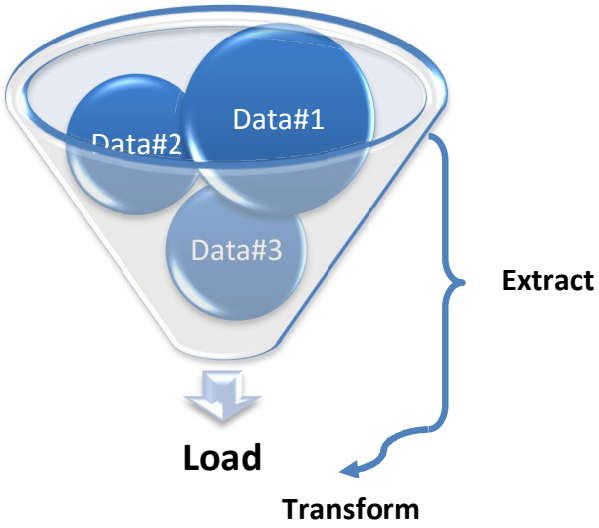


Fig 16 Illustrative concept of the ETL process cascade. The funnel represents the transformation step
Data #1..#n are heterogeneous data sources

[12] ftp://ftp.arabidopsis.org/

warehousing (BA Devlin, 1988) is commonly used.

> *"Premature optimization is the root of all evil", by Donald Knuth*

Extraction often implies conversion of the relevant data from different sources into one common format, and can be the most challenging step in the ETL-process cascade. *Transformation* comprises tasks such as missing data imputation, deriving values, joining or splitting values, and bringing the data into a format for the *data loading* into the devised *data schema* which depends upon the designated purpose of the data. Following the creation of a schema definition, the data is then imported.
Empirically speaking, ETL may resembles an E(TL)^n process, wherein the *transform and load* process repeats several times through an iterative refinement and testing-period, until the database-schema is such that it optimally captures the imported data. In practice, the resulting SQL statements translate to an initial CREATE TABLE and a repeated sequence of (LOAD DATA, TRUNCATE TABLE, ALTER TABLE)^n statements.
ETL tools can significantly speed up this process.

For instance, to load the KEGG Arabidopsis th. pathway-map accession-list containing map-ID numbers and the pathway names, a regular Expression is written and executed on the cross-platform *MySQL Query Browser*[13]tool *(Fig 17)*.

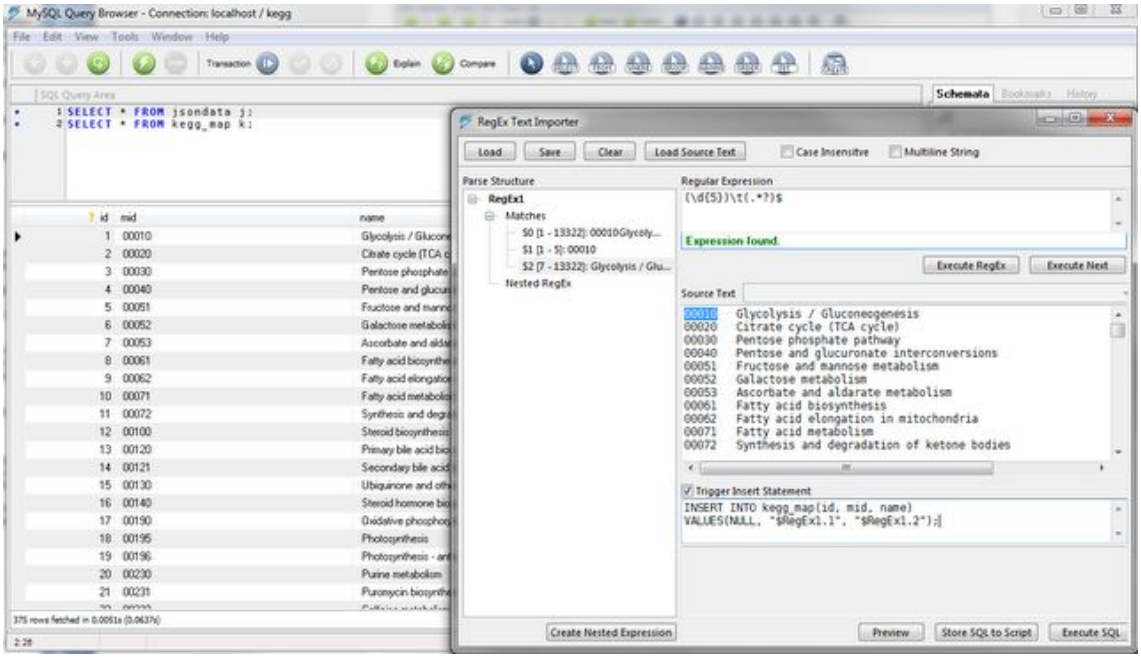


Fig 17 Using the undocumented RegEx Text Import Feature of the *MySQL Query Browser* v1.2.17 to perform simple ETL tasks. The Regular Expression - `(\d{5})\t(.*?)$` , matches five digits, separated by a tabulator and a second match of any characters until a new line or line-feed character occurs. Each line matches is transformed into 'INSERT INTO…' SQL statements- to insert the matched values into a pre-created table-schema.

[13] http://dev.mysql.com/doc/query-browser/en/index.html ; no longer supported http://dev.mysql.com/support/eol-notice.html

→An example of an ETL Script for extracting web resources is provided in the Appendix

2.5 Data-Sources: Biological Databases

Following is a concise list and description of biological databases, used as primary data sources for the status quo of the Aggregator portal.

2.5.1 KEGG - Kyoto Encyclopedia of Genes and Genomes

KEGG is an online database providing genomic, enzymatic, pathway and biochemical data (Kanehisa, 1997). KEGG initially consisted out of five databases, KEGG Atlas, KEGG Pathway, KEGG Genes, KEGG Ligand, KEGG BRITE. BRITE[14] lists *functional hierarchies and binary relationships of biological entities*. The different KEGG databases are highly integrated to provide a digital representation of the biological system. Additionally KEGG provides an increasing range of text-mined information, which is initially set to be outside the scope of the Aggregator.

Within the project's scope, KEGG provides information on molecular interaction networks, such as pathways, compounds and enzymes.
Beginning with July 2011, KEGG switched to a subscription model for direct FTP access to the databases. Full database access to up-to-date database entries is still provided via the *dbget* interface. To get a web-rendition of the content, for instance for the *Primary bile acid biosynthesis* (map00120) the following url can be called:

http://www.genome.jp/dbget-bin/www_bget?map00120.

→KEGG provides a SOAP API. The KEGG API is fully mapped to the web-portal (See 2.2.1, 3.5.2.5, 3.5.2.6).

2.5.2 BioMeta Database / KEGG Ligand

BioMeta is a database of metabolites and metabolic reactions. Its contents are largely based on the KEGG Ligand database. Compared to the KEGG database, a large number of chemical structures have been corrected with respect to constitution and stereochemistry. The database aims to provide correct representations of molecules and reactions. Roughly 1,500 molecular structures have been corrected and about 55% of the unbalanced reactions have been "balanced". (MA Ott, 2006)
Four *flatfiles* are provided *rxnflat.txt*, *molflat.txt*, *srcflat.txt* and *enzflat.txt* which are dated to June 2007, with entries separated by *<entry>* - tags, and line-content is denoted by a two character code, which was converted into a SQL file via a previously created Python script (Sauer, 2011) and custom written Regular Expressions.

[14] http://www.genome.jp/kegg/brite.html

Biometa is used for cross-referencing of reactions for the data-aggregates.

→BioMeta is federated into a local database.

2.5.3 ExplorEnz – A MySQL database of the IUBMB enzyme nomenclature

The database ExplorEnz, is the primary repository for EC numbers and enzyme data, curated on behalf of the IUBMB. The enzyme nomenclature is incorporated into many other resources, including the ExPASy-ENZYME, BRENDA and KEGG bioinformatics databases. As a MySQL database, the formatting of chemical and enzyme names is preserved and allows simple as well as complex query constructs. The database is available for free download as SQL and XML file at http://www.enzyme-database.org (AG McDonald, 2007).
ExplorEnz was directly federated into a local Aggregator database. It is used for enzyme information, links to other databases, enzyme classes and primary literature references.

→ExplorEnz is federated into a local database.

2.5.4 TAIR[15]

The Arabidopsis Information Resource is a web-portal for genomic and, to some extends, proteomic data for Arabidopsis thaliana. TAIR provides information about genes, markers, polymorphisms, maps, sequences, clones, DNA and seed stocks, gene families and proteins and links to other databases (Garcia-Hernandez M, 2002).

AraCyc, for visualizing biochemical pathways was developed as part of TAIR and is now incorporated in PMN (Zhang, 2005).

TAIR flatfile databases have been acquired from the TAIR website and federated into a local database via several ETL processes. The datasets provide the aggregator with Arabdiopsis specific pathway, reaction and metabolite information.

→TAIR provides a REST interface. Additional information is provided via link-outs.

[15] http://arabidopsis.org

2.5.5 PROMEX - the mass spectral reference database

ProMex provides spectral information linked to experimental metadata such as species, tissue, organ, treatment and the Mass-spectrometry instrument type (e.g. Orbitrap, LTQ, TSQ). The library initially contained 4557 manually validated experimental mass spectra (Hummel J, 2007). Recently ProMex was revised and implemented as an AJAX web-application (V. Egelhofer, publication pending).
Users can query and retrieve proteomics data via protein accession numbers, protein names, organism name, peptide sequences and spectra.

The current ProMex version is built around a process-flow of three successive JSON queries to retrieve a spectrum, which were compounded into one query to allow spectra data preview. Detailed information will be provided via a *link-out*.

→ProMex provides a JSON interface. Within the scope of the Aggregator, Data is retrieved by a Protein Accession numbers (‘*mgate_getPromex’*).

2.5.6 SUBA II - the Arabidopsis Subcellular Database

SUBA provides experimental information of protein localization to facilitate understanding of protein function and of biological interrelationships. At least 1100 related experiments completed in Arabidopsis are referenced. Proteomic surveys of subcellular components in Arabidopsis are available for roughly 2600 proteins. (JL Heazlewood, 2007)

→ SUBA provides a JSON-RPC interface. Within the scope of the Aggregator, Data is retrieved by passing one or multiple Arabidopsis Gene Accession numbers and data is used for graphical representation of the localization of enzymes. Details are provided via a link-out to the *SUBA flatfile-entry*.

2.5.7 Massbank

MassBank is the first public repository of mass spectral data for sharing them among scientific research community. MassBank data are useful for the chemical identification and structure elucidation of chemical comounds detected by mass spectrometry (H. Horai, 2010).

→Massbank provides a SOAP interface. At the moment, the task of cross-referencing local entries with Massbank Accession Id’s is facilitated by extracting Massbank accession Ids from the website through a DOM-parser.

2.5.8 AraCyc / PlantCyc[16]

The Plant Metabolic Network (PMN) provides a broad network of plant metabolic pathway databases that contain curated information from the literature and computational analyses about the genes, enzymes, compounds, reactions, and pathways involved in primary and secondary metabolism in plants. (Rhee SY, 2005)

The Aracyc datasets are reflected in the TAIR datasets. See 2.5.4. Additionally Reaction information is linked-out to the Aracyc databse. The TAIR datasets provide Reaction-Accession Ids.

2.5.9 RIKEN PRIMe and Omicspace

The RIKEN Institute of Physical and Chemical Research in Japan provide many databases which are accessible via a Hub-Site, named RIKEN Omicspace[17], which is integrated in the aggregator portal for biomarker information and to furnish a research overview on a particular compound.
RIKEN PRIMe, the Platform for RIKEN Metabolomics, is a Web-based service for metabolomics and transcriptomics providing multi-dimensional NMR spectroscopy, GC/MS, LC/MS, and CE/MS spectra as well as related tools (Akiyama K, 2008). PRIMe is only integrated within the Aggregator at a preliminary web-service level. Most spectra are available through Massbank, which is fully integrated in the aggregator.

→ No web-interface API for PRIMe is available. 13C-HSQC Spectrum peaks, of less than 100 compounds, have been web-scraped and parsed into a flatfile database.

2.5.10 Other Databases

ChEBI or Chemical Entities of Biological Interest[18] is a freely available dictionary of molecular entities focused on 'small' chemical compounds. ChEBI provides group information, classifications and ontologies with relationships information between molecular entities or classes of entities and their parents and/or children (K Degtyarenko, 2007). ChEBI provides a SOAP interface use to augment chemical information such as chemical relatedness and similarity within the Aggregator.
Pubchem[19] is only used for similarity and chemical structures, despite providing the most feature rich information in terms of biological activity of compounds. The Pub-

[16] http://www.plantcyc.org/
[17] http://omicspace.riken.jp/
[18] http://www.ebi.ac.uk/chebi/ - ChEBI
[19] http://pubchem.ncbi.nlm.nih.gov - PubChem

Chem Compound Database contains validated chemical depiction information provided to describe substances in PubChem Substance. Structures stored within PubChem Compounds are pre-clustered and cross-referenced by identity and similarity groups (Bolton E, 2008) SOAP, RESTful APIs are provided ('PUG-API'), in addition to full database downloads. Only the PubChem SOAP-API is used via the provided WSDL service-map[20] and fully mapped to the Aggregator Client via a JSON-RPC (See 3.5.2.5).
The *KNApSAcK* Database contains a Comprehensive Species-Metabolite Relationship Database (FM Afendi, 2011), and as part of the cross-link annotation of the TAIR datasets is provided in the form of link-outs, within the scope of the Aggregator. The last update was 2008. No web-API or download is provided.
The National Cancer Institute Databases (*NCI-DB*) provides various cheminformatic services and information for many compounds.
CAS, by the Chemical Abstracts Service, provides the largest and most comprehensive database of chemical substance information, and is an official chemical registry.
The *PathLocDB*[21] –Database provides pathway localization information, subcellular and multiple localizations of pathways summarized from UniProt and KEGG pathway databases (Min Zhao, 2011). The provided flatfiles were transformed and loaded into a local federated database. *Publichouse* is a large federated Biological Datawarehouse and provides supportive information such as superpathway-clusterings.
Aracyc is used for Pathway and reaction information for Arabidopsis thaliana and *Plantmetabolomics.org* provides Metabolomics datasets delineating Environmental from Genetic influence based on experimentally set genotypes and environments.
An overview of the databases, APIs, and steps performed is provided in Tab 2.5.

Tab 2.5 - Metabolomics related data sources served by the Aggregator web-portal
(highlighted: functionally mapped but not integrated in the GUI), underlined: primary use

Name	*Link*	*Description*	*API*	*Reference*
SRI Public-House	biowareh-ouse.ai.sri.com	DataMart, PH v15.0	✓	(T J Lee, 2006)
TAIR	arabidopsis.org	Genome annotation		(Swarbreck D, 2008)
SUBA	su-ba.plantenergy.uwa.edu.au	Subcellular localization (MS, FP)	✓	(JL Heazlewood, 2007)
Aracyc/PMN	plantcyc.org	Pathway, Ontologies, Reactions	~	(Akiyama K, 2008) (Zhang, 2005)
KEGG	www.kegg.jp	Pathway, Enzyme information, Ontologies	✓	(Kanehisa M, 2000)

[20] http://pubchem.ncbi.nlm.nih.gov/pug_soap/pug_soap.cgi?wsdl
[21] pathloc.cbi.pku.edu.cn/

		(KO's), Reactions		
Massbank	www.massbank.jp	MS/MS, Others	✓	(H. Horai, 2010)
ChEBI	www.ebi.ac.uk/chebi/	Small molecules, ontologies, relations, similarity	✓	(K Degtyarenko, 2007)
Pubchem	pubchem.ncbi.nlm.nih.gov	Compounds, activity, physical, physiological, similarity, literature, ontologies	✓	(Bolton E, 2008)
PathlocDB	pathloc.cbi.pku.edu.cn/	Sub/cellular Pathway localization	-	(Min Zhao, 2011)
KNApSAcK	kanaya.aist-nara.ac.jp/KNApSAcK	Metabolomic compound information	-	(FM Afendi, 2011)
ExplorEnz	enzyme-database.org	EC classification as SQL dump	-	(AG McDonald, 2007)
CAS	www.cas.org	Official chemical register	✓	(ACS, 2012)
NCI DB	http://cactus.nci.nih.gov/ncidb2.1/	Chemical web-services, compound info, daylight canonical SMILES	✓	(NIH, 2012)
PlantMetabolomics.org	plantmetabolomics.org	MS/MS spectra, processed, lipidomics, gen/env-ratio	-	(Bais, 2010)
ProMex	www.promexdb.org	MS/MS spectra	~	(Hummel J, 2007)
PRIMe	prime.psc.riken.jp	MS/MS, NMR spectra, Others	-	(Akiyama K, 2008)

2.5.11 Ontologies

In order to simplify data handling in terms of manipulation, comparison and cross-referencing of data-objects/entities, controlled vocabularies or *ontologies* are often used. The SUBA database uses AmiGO (Carbon S, 2009). Gene Ontology (GO) (Ashburner M, 2000) and the Plant Ontology (PO) (Avraham S, 2008) are used within the project where applicable.

3. Aggregator Implementation

This chapter briefly elaborates the issue of data collection, and describes the target audience of the web-portal, and the implementation of the system design prototype.

3.1 Data Aggregation

Aggregation is the process by which a compound object is built out of several smaller ones (J Heaton, 2002). In a practical example, a *KEGG* chemical compound entry is retrieved through its exact mass plus a certain mass tolerance and the subsequently returned top-compounds are collated with entries from a local database to be formulated into a compound dataset containing *ChEBI* ontological information and *Pubchem* biological activity and chemical similarity information. An additional characteristic during formulating mixed scientific datasets, is keeping track of the primary literature that encompasses each respective piece of data. Literature information should be tracked, collated and included in the compounded dataset for reference and future verification of the aggregated information. This process, wherein data is added and restructured by a data handler, is referred to as *Mediator based Integration*, whereas if only links or URI (universal resource identifiers) are included, resulting in no or few collation tasks, the term *Navigator based Integration* applies (Hernandez T., 2004). Mediator or Navigator refers to the level at which the source/target combining occurs. Seligman et al. lay out a workflow of eight steps for successful data integration, beginning at gaining *basic* and *semantic knowledge* about the data source and its application, creating and specifying *data- transformations*, *combination* and *mapping rules*, *cleaning* and *loading* the data and finally the creation of a user-friendly access environment or *GUI* (L Seligman, 2002). Two approaches towards data provision are taken in the Aggregator, differing in the nature of the data-curator and therewith the provisioning capabilities. The following sub-chapter (3.1.1) will briefly discern manual from automated or non-manual data provision (3.1.2).

3.1.1 Manual Data Provision

One of the main facets, inherent to modern biology today, is the collection of large amounts of data that delineate biological functions and interactions at different hierarchical levels, be they of molecular or global scale. Often summarized information can be found in publication supplements of Omics-related projects, with large amount of additional data being offered in project-related databases established for the pur-

pose of publishing large datasets. The wide range of data formats and the frequency of data provision of different projects raise the need for automated data aggregation. However, sometimes data providers desire or need to deliver their information manually. Reasons may involve technical barriers, preliminary or limited information, unexpected data fixes and/or updates where the data authority does not want to submit changes immediately to the database-system without waiting for the next update-cycle. Additionally, biological information is often presented in semi-structured text-files which require manual steps such as locating the data-source, ETL tasks and database updates.

3.1.2 Automated Data Provision

Usually, aggregation of biological information is carried out automatically and regularly via data collection technology. The non-manual data provision may prove to be appropriate by the simple fact that insufficient quality or quantity of data may procure retribution or indifference from the scientific community. Automation is able to eliminate the risk of human errors in data provision. Once a user or data-consumer requires some data, the system automatically collects and provides data without human interaction. Such a service may be installed, monitored and configured by each single data provider. Practically a division of labor is advantageous. Therefore data aggregation and data generation by separate instances allows single data providers to take better care of curating and provisioning their data without wasting resource on data integration. This model shows less uncertainty overall, in terms of research funding and project support (HJ Joshi, 2011; D Reynolds, 2004).
In practice, the aggregator frequently consolidates distributed web-services into a central service-function on the basis of a common identity, such as a chemical compound. For instance, the service function *RxnTAIRtoKEGG* , when passed compound names, returns the best matching reactions from TAIR (2.5.4), along with an image of the reaction equation:

```
rpc.get_RxnTAIRtoKEGG('sulfite + thiocyanate')
```

The response of the invoked web-service is then a JSON-object as follows: (Note: the image-data is truncated for brevity):

```
XHR finished loading: "http://localhost:8888/gator/rpc_handler.php".
Result id#22:  rpcClient.js:718
[Object
ImageRxn: ".../////=="
compound_ids: ["C01755", "C01326", "C00094", "C00080", "C00320"]
rxn_id: "R01930"
uid: "627"]
```

3.2 Web Services

A web service is a software system designed to support interoperable machine-to-machine interaction over a network (W3C, 2004).

Web services typically provide an interface to allow access to application functionality by using the HTTP protocol. A typical web service workflow consists of a request by a *client* or *service-consumer* to a *server* or *service-provider*, followed by a response from the server, containing a unique-message flow ID (See the preceding example in 3.1.1). The data format and underlying protocols for the request/response, depend on the service-provider. The most common types of web services are SOAP-based, JSON-based and RESTful. As a rule of thumb, complex services are more suitable for SOAP, and simple services more suitable for JSON, REST or RESTful API's. All communication is wrapped in an HTTP-header, with the exception of web-socket traffic as part of an experimental communication interface of the JSON-RPC provided by the aggregator toolkit (See 5.52.5; Websockets are not discussed due to their changing RFC-Draft status).

3.2.1 SOAP - Simple Object Access Protocol

SOAP -based web services ("Big Web Services") use an XML container as interoperable messaging format, which requires an underlying XML parser, therewith enabling complex content verification and extraction capabilities. After the XML-document definition, SOAP includes three elements, comprising an *envelope*, a *header*, and a *body* (S Weerawarana, 2005). Web-service functions are summarized and described in an XML-file termed Web Services Description Language (WSDL). These files are cached on the server in the directory *./res/data* with the file-extension *.wsdl* and the WSDL file-name determines the *service-domain* of WSDL-described service-functions. Service-domains may be loaded on-demand in the Aggregator-client. Individual SOAP functions can be extended ('overwritten') in the aggregator service-files, simply by *redeclaring* the function with a new function routine. More information on using the Aggregator web-services is provided in the Appendix and in 3.5.2.5, 3.5.2.6.
An analog to WSDL for JSON-RPCs is realized in the form of extended Service Map Descriptions or SMDs, and are used internally for the Aggregator-server and its modules.

3.2.2 REST and RESTful web-services

REpresentational State Transfer (RESTful) web services directly utilize the HTTP protocol and its header rather than specifying the use of additional message wrappers. REST is an architectural style, which is similar to the Internet, where each resource such as a

web-site or an image is uniquely addressable (L Richardson, 2007). Thus RESTful web services are resource-oriented, unlike the function-oriented SOAP protocol. Re sources are resolved by URIs. Tab 3.2 compares the canonical Create, Read, Update and Delete (CRUD) operations in SQL to stateless CRUD in HTTP and REST.
To easily integrate RESTful services at the server-side, a *Rest_Service* class was implemented in the programming language *PHP* to allow for object-oriented programming by treating resource-URIs as objects, assigned to properties such as service-function-names.

Tab 3.2 Comparison of CRUD Operation in REST vs SQL

Operation	SQL	HTTP/REST
Create	**INSERT**	**POST**
Read (Retrieve)	**SELECT**	**GET**
Update	**UPDATE**	**PUT**
Delete (Destroy)	**DELETE**	**DELETE**

3.3 Data Integration Challenges

Furnishing a realistic time-schedule for data-related tasks prior to implementing an aggregator is advisable. Thus understanding the intricacies of data integration is important when considering data federation of Omics resources. *A. Li*, summarizes the challenges in molecular biology databases in seven points as *highly heterogeneous data*, *large data volume with unique data types*, *highly dynamic data sources*, *highly hierarchical data* - by nature, *lack of standards* and *ontologies* (controlled vocabularies), *Infantile DBMS and data access tools* and hypertext yielding *ambiguity in identifiers* (A Li, 2006).

Empirically, the main challenges of data federation faced during the creation of the Aggregator project are widely congruent to the aforementioned aspects, but can be roughly divided into challenges of technical nature and challenges of semantic nature, both of which impede human accessibility to information.
The key challenges are locating and keeping track of the myriad of project-relevant data providers, the many data formats, many access languages, lack of semantic integration, lack of publicly and centrally compiled cross-reference database-identifier information and non-disclosed data.
The foremost technical challenges faced throughout the project, are large, complex, incomplete datasets and writing efficient database queries which collate many entries.

3.4 Users and usage scenarios

> ***"We've spent decades studying metabolism, but ironically very little of this has been brought to a diagnostic application. You could say that metabolism is a***

mature science looking for a game to play in".

Bruce German, University of California, Davis

A web-aggregator portal for Metabolomics is important for efficient access to heterogeneous, metabolism-related data. Current data portals provide solutions, but with limited practical value, as the requirements from usage scenarios are rarely taken into account.

The targeted audiences for the aggregator portal are plant biologists and biochemists. As such many portal-functions require the comprehensive Arabidopsis datasets to be fully operational. The aggregator allows switching the organism around which data is centered, based on the KEGG list of organisms. Such action consequently results in partially non-operational functions which are grayed-out in the user-interface, facilitated by a special-purpose warning passed along the message envelope of the server response, to which the user interface reacts. Much of the initial aggregator concept is built around the coexistence of a local federated Arabidopsis Integrated Database for Metabolomics and external web-services which typically offer a wider biological scope.

Furnishing a clear picture of the future usage-scenarios of an application is important to the development cycle, presentation aspects, the user-workflow, and the provisioning of data and web-services. Ultimately, the targeted scientific community should benefit from a tool that adds genuine value to the vast repertoire of existing biological utensils.

The difficulty of delineating usage scenarios for the web-portal stems in part from the field of Metabolomics itself, which is less clear-cut than proteomics and genomics. Still without consensus in the metabolomics community, is the question of “How do we deal with data that don’t make biological sense based on literature and common knowledge?” (U Roessner, 2009) In practice data-ambiguity impedes the development of presentation scenarios.

Fig 18 Primary layouts serve the user and developer with less information-clutter and concurrently retrieved information by quickly turning off or on the list of data providers to be queried. The Layout selection is not shown by default.

To cope with an expanding list of data providers and uncertain usage scenarios, three primary user-layouts were devised. These are *Biomarkers*, *Metabolomics*, and *Systems*

Biology (Fig 18), which can be selected by the user to preset different levels of information, deemed existential to those three fields. All future user-interface functionality is ascribed to one of these layouts either automatically through regular expression matching of 'tab'-titles or explicitly. *Universal* in Fig 18 is the preset default-layout and which does not omit any information. By providing an option to quickly omit unnecessary information, the site is easier and more responsive to navigate, the server takes fewer loads, and an overall faster user experience can be delivered, in addition to the same advantages applying to portal-developers.

Interesting applications for data-integrative Metabolomics are Biomarker Identification, generating research hypothesis or validating hypotheses, bridging the gap between genotype and phenotype (O Fiehn, 2002) through data mining of metabolomic data sets and their corresponding metadata, bridging the gap between environment and genotype (*Plantmetabolomics.org* Datasets) as well as taxonomy and phylogeny related application scenarios, through revelatory 'hallmark' secondary metabolites (F Pietra, 2002).

It remains to be seen how many of those fields, suggested in the user-layouts (Fig 18), can potentially benefit from the Aggregator portal and in what manner.

Concluding, whilst the aggregator does not attempt to handle the daunting task of deriving biological sense from measured data-sets, it can attempt to provide a gateway to the data and underlying literature and overall knowledge that aids in hypothesis creation.

3.5 Implementation

3.5.0 Recapitulation

The Arabidopsis community is rich in online resources. Often metabolomic studies lead to the creation of online resources (Weckwerth W, 2008). Locating and navigating these resources can be difficult (HJ Joshi, 2011). Recently the MASCP-gator portal[22] was established by MASC, to offer a summary-aggregation portal to retrieve proteomics data-resources (HJ Joshi, 2011). An integrative summarization portal for Metabolomics, bringing together aspects of chemistry, biochemistry and systems biology is not yet found in the literature, but the specific need of such stated (E Schwarz, 2012).

[22] http://gator.masc-proteomics.org/

3.5.1 Objective

Following are key aspects *a)* to *f)* which set the objectives for the project development. An aggregation portal for Metabolomics should a) summarize and b) align data for presentation, within c) a coherent user interface. c) Search types may be Compounds, Reactions, Proteins/Enzymes, Genes, Spectra/Peaklists, Pathways and Keywords or Ontologies (controlled vocabularies). d) The aspect that genes correlate less with metabolites than genes do with proteins (U Roessner, 2009), should be heeded in a presentation context. e) Novel strategies for visualization should be developed to help communication and interpretation of scientific data (W Cleveland, 1985).

Due to the scale of such an enterprise with an incremental-development outlook, f)emphasis must be put on a future-ready architecture. The architecture must be sustainable over a long time-period, and reusable or modular and maintainable by a disparate group of developers. Future-readiness in a Metabolomics-context requires data and computing scalability, should at some point Metabolomics data-sets be processed. Meeting this requirements are Service Oriented Architectures (JY Chung, 2005). Indeed, technical challenges pointed out in 3.3 can be met with resource distribution.

3.5.2 Architecture

Service-oriented computing promotes the idea of assembling application components into a network of services that can be loosely coupled to create flexible, dynamic business processes and agile applications that span organizations and computing platforms (MP Papazolou, 2007).

3.5.2.1 State of the art: Distributed SOA web-applications

The state-of-the-art for many biologists is to employ service oriented scientific workflow approaches to high-throughput experimental data (A Manjunatha, 2010). Service-oriented workflows use Web services as the atomic building blocks (B. Youakim, 2008). Recently, NMR-based Metabolomics has seen successful application of service oriented workflows in data analysis (A. Manjunatha, 2010). In contrast to classical centralized SOA architectures, distributed SOA can connect any distributed application without restriction to support data and function sharing, whilst affording scalability, robustness and availability (Z Wu, 2009)

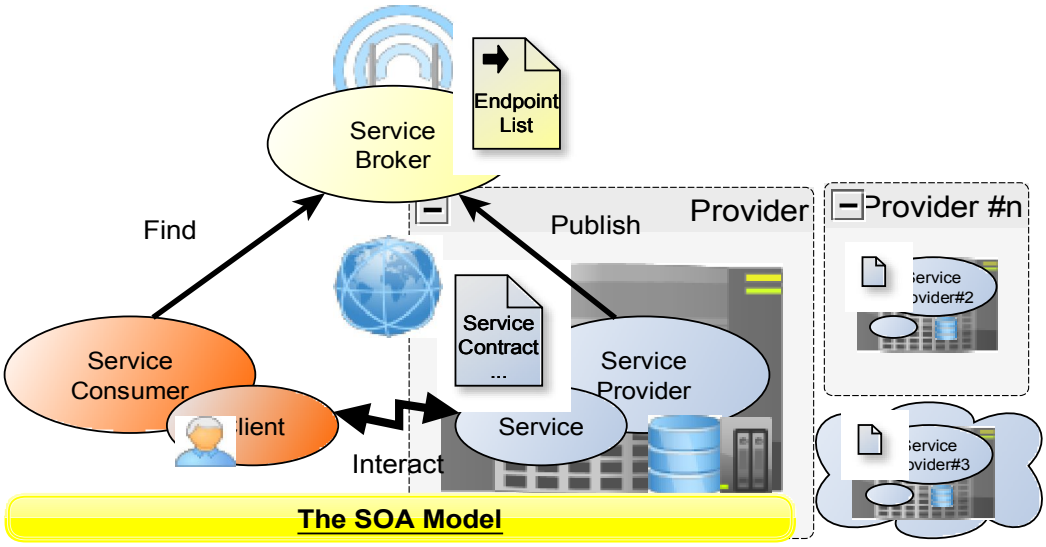


Fig 19 Showing the distributed SOA architecture of the Aggregator. A Client acts as the Service-Consumer, initially requesting a list of endpoint-providers from a Service-broker. This broker can be located at a Provider instance or externally. Upon receiving the list of endpoints, the Client chooses and connects to a Provider-instance, which may be a simple web-servers or a virtual server within a cloud environment. Subsequently the Service provider provides a Service contract (SMD-file), allowing the Client to map remote functions and to start invoking service-functionality through the web-application.

Adapted from H.Haas – SOA architecture, W3C (H Haas, 2003) – Depending on the SOA Model definition, the Service broker may be optional.

Fig 19 demonstrates the basic distributed SOA architecture applicable to the Aggregator implementation. Additionally, the endpoint-addresses can be assigned to individual service-functions rather than entire service-providers or instances, thus varying the *endpoint-granularity*. The connecting Client first requests a list of endpoint-providers from a *service-broker* (See 3.5.2.3), whereupon the client application chooses a provider at a specific endpoint-address, thus rendering the web-application operational.

3.5.2.2 Distributed aspects

Rather than a centralized repository, scientific data is generated and hosted by various projects with individual funding and individual data curators (Fig 20). Static or frequently accessed information is consolidated into a local database, which is directly connected to the server through a low latency interface (*libmysqlclient* or *ODBC*). New external API-providing data services at different endpoints are easily incorporated by creating and loading service-maps of the target project. The distributed data model is more robust towards funding and project support (HJ Joshi, 2011). Additionally, an operationally sound distributed data model should allow focusing on the presentation of the data, as opposed to focusing on data-federation schemas (KD Schewe, 2008).

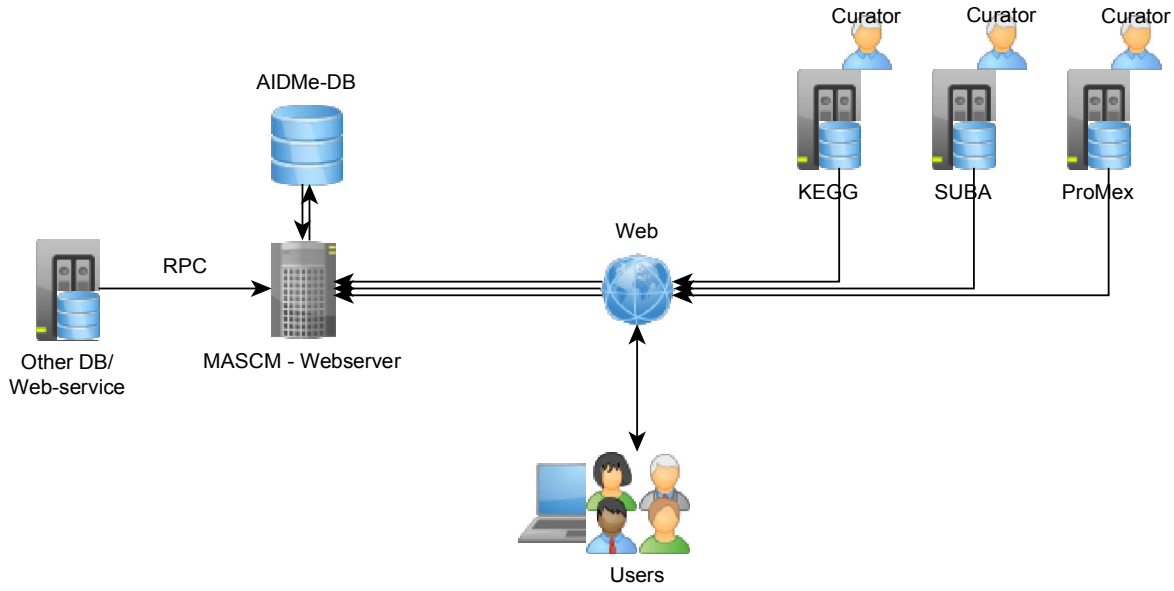


Fig 20 Aggregator Network Topology Overview (Single Server) for a potential application scenario

Individual curators maintain individually financed, public projects which are accessible via the Web (KEGG, SUBA, ProMex, etc.). This distributed data model shows to be more robust towards funding and project support (HJ Joshi, 2011). Users can access the webserver ('MASCM'… a potential project designation) in form of a web-application portal. Other Databases or web-services can be easily integrated into the portal (See 3.5.2).
The Arabidopsis Integrated Database for Metabolomics (*AIDMe*) is a project-related, federated Database directly connected to a server through a low latency interface.
DB….Database
RPC...Remote Procedure Call (remote access of a application functionality)
'Web'…abstraction/summarization for ISPs, routing and general internet-network-topology

3.5.2.3 The service broker

The service broker may be located externally or at the server-instance. With *#n* server-instances, at least *#n-1* endpoints would have to be present in the list for distribution of services to different endpoints. If no list is provided, distributed aspects do not apply, yet the services and client-application remains operational. An external service broker could be at any address-location that ensures high-availability, OAuth protocol support, and limited CRUD-operation capabilities, such as twitter-*channels* or even google-*webpages*. In a concrete example, a server-instance posts its status to a service-broker every full hour. In case of public broker channels, full transparency is guaranteed, which can be a demanded aspect in biological sciences. Connecting clients directly download a feed of the status-messages posted over the past hour. Instances with less load or more computing resources may post more frequently within the designated update timeslot, thus increasing the chance of being connected to. When transparency of the broker is desirable and service-endpoint granularity not needed,

this concept is preferable and is implemented as a proof of concept. Upon initially connecting, the Client randomly picks an endpoint from the list of service-endpoint-providers to distribute the overall load. Should an endpoint not respond within a brief timeout-window, the endpoint is excluded and another endpoint randomly picked. A timeout does not discern between network latencies and computing latencies, but rather summarizes both.

3.5.2.3 The physical server unit

In order for the client application to be operational, underlying web-services must be provisioned at a server, requiring upfront considerations towards computing-demand and the overall costs per fiscal period. Physical provisioning of computing and networking hardware can be delegated by using a cloud-provider such Amazon's EC2[23]. Alternatively web-server providers such as internal academic networking environment of a university, any other commercial web-host or even free LAMP (Linux-Apache-MySQL-PHP) server offerings[24] can be used. To this end, the project's underlying toolkit aimed to support the common denominator feature-set provided by most LAMP servers. The Cloud, allows pooling computing resources from clusters of servers and dynamically assigning or reassigning virtual resources (Q Zhang, 2010), with easily assessable, metered costs.

Generally speaking, server architecture and server applications are highly modular, allowing physical migration and parallelization with growing demand. Thus, for instance, offline migration of the *Relational Database Management System* (RDBMS) is easy. Some virtual servers allow migration of the entire booted operating system in a live state to different underlying physical hardware. Hence program-developers can migrate their own virtual Operating System (OS) image seamlessly, saving configuration time and costs.

Planning the required server capacities is described by M. Bichler et al (M Bichler, 2006).

[23] http://aws.amazon.com/ec2/ Amazon Elastic Compute Cloud (Amazon EC2) ; http://www.howtoforge.com/how-to-create-a-lamp-setup-apache-2-php-mysql-on-centos-5.x-in-an-amazon-linux-ami Tutorial for creating a LAMP OS image,

[24] http://www.free-webhosts.com/free-php-webhosting.php Free PHP Web Hosting, DOR: 11/2011

3.5.2.4 The logical server unit

As shown in Fig 21, a logical server, assumed as a 'black-box' with an IP-address and input/output pins to the Web, is comprised out of a) a web-server with a low latency CGI interface to one or several dynamic-programming languages, b) a Relational Database Management System (RDBMS), c) the Operating System (OS/Kernel) and a d) POSIX conforming console environment.
Several logical units make up the distributed computing environment, with each unit constituting an addressable-resources or endpoint on the Web. Each unit is fitted with the same version of a modular software package or *'toolkit'*, introduced in the following *Chapter 3.5.2.5*.

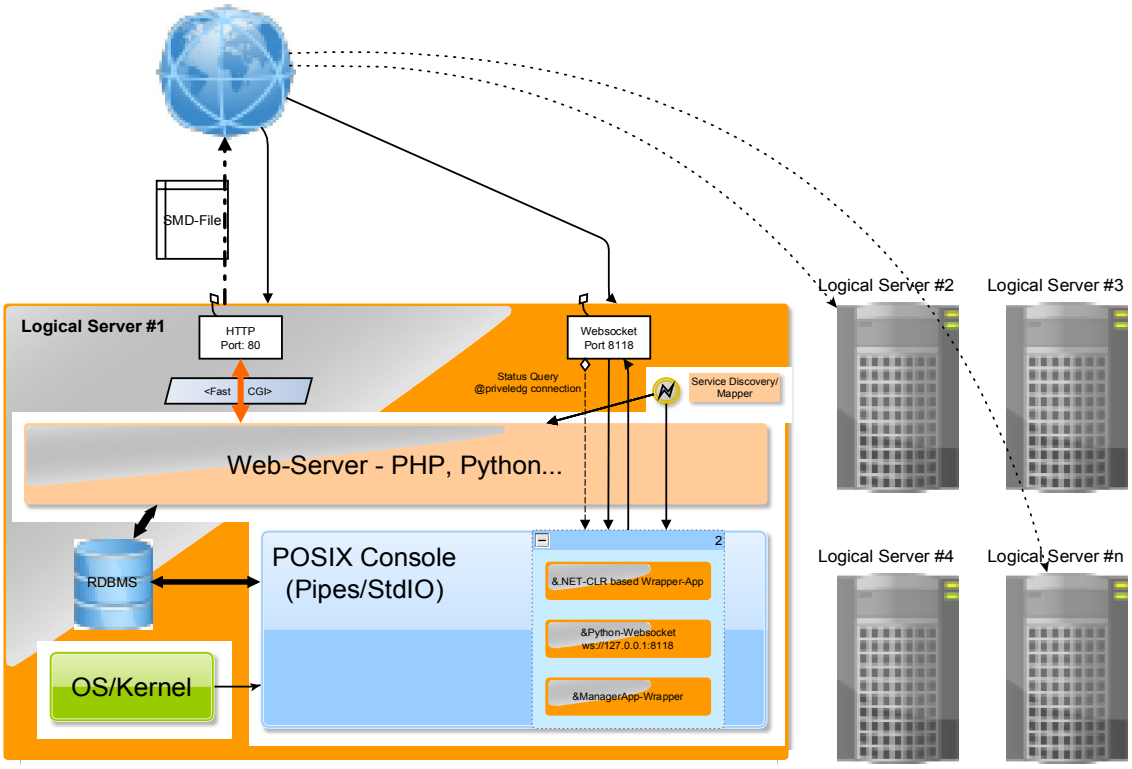


Fig 21 Displaying the logical Architecture of a server-unit hosting an aggregator-instance. Several units comprise the distributed computing environment. Computationally intensive services or functions may be hosted in a Cloud environment, thus delegating responsibility towards provisioning hardware to meet current resource-demand.
Initiation process flow:

1. **The Client requests a service-map(SMD) via HTTP or websockets.**
2. **The Server Unit requests ('discovers') the services currently provided by a) the CGI-script, b) background processes at specific ports, c) POSIX-command programs, which are summarized ('discovered') by a script**
3. **The Client loads the SMD and the web-application becomes operational**

Only a particular set of service functions, for instance functions related to spectra-processing, may be outsourced to a Grid or Cloud computing provider. Outsourcing functionality is performed by adding an optional endpoint-entry of the targeted function or set of functions to the Service Discovery and Mapping (SMD)-files. The SMD-file also serves as a *Service-Contract* as shown in Fig 19.

3.5.2.5 The Toolkit : Rapid prototyping of scientific SOA web-applications

The Toolkit consists out of a set of small libraries, some of which are optional and offer rapid application development (*RAD*) and deployment of scientific web-applications, which heed the *MVC*-model and can be setup in a distributed *SOA* environment. Following, these claims shall be dissected.

Rapid application prototyping is facilitated by a short learning period required to familiarize with the small codebase of the toolkit, only consisting out of a robust client-RPC interface library, a Service Discovery module, a service-handler and a service-template for implementing own web-services with additional web-resources such as design-templates, application-examples, graphics and stylesheets.
For the SOA web-application concept to work, server-side functions must be exposed, and mapped to user-interface elements on the web-application. The idea of the toolkit is based *firstly*, on the developer either using the user interface elements defined in HTML5 or choosing a lightweight web-user-interface library with which the developer is already familiar. Such a library may be jQueryUI, but for basic application building the toolkit-supplied templates and designs are likely to suffice.
Secondly, web-services are programmed as simple functions without heeding additional syntax. The toolkit's discovery and handler-script takes care of compiling a list of functions into a Service-map. Service-Function 'Manuals' can be added and mapped to the Client through HereDoc-Styled Comments (`/** Comment */`) prior to the function's declaration. Alternatively, a *manpage* providing help for a functions within a given service-domain can be put into the *res/data/* folder, for instance *kegg.man*.
These features enable quick lookup of function-descriptions directly in the Browser-environment, which become increasingly important to the developing-cycle and thus aid rapid development. The distribution-aspects (3.5.2.2) are seamless and a matter of the existence of a list with service-provider endpoints.
The toolkit allows putting emphasis during development, on either the Server-side or the Client side. This emphasis-model could be expanded to two developers efficiently working in tandem on the server aspects and client aspects, respectively. Client-Server cross-development is facilitated by seamlessly bringing an up-to-date or incrementally updated pool of federated web-service functions directly to the Client in the form of an RPC *object*. Thus a client-developer can work directly within the same Browser-environment as a user would when using a toolkit-based web-application. The RPC maps verbose message-feedback such as notices and warnings from the server-environment directly to the Browser's JavaScript environment.

JavaScript code implementations may be shifted from the client to the server and vice versa by using server side JavaScript engines such as Node.js (Joyent Inc, 2011) allowing leeway during performance critical development code-stages.

To specifically build Aggregators for scientific data, the existing codebase of web-services may be used.
Finally, the MVC adherence aspect of the toolkit is elaborated through a small code example, which also serves to demonstrate the aspect of rapid prototyping (Fig 22).

```
<script src="res/js/rpcClient.js" iRPCautoLoad="default"></script>
<div><b>Retrieve XRef Info for a given Compound... </b><br><input/>
<button onclick="rpc.get_xrefByName(this.previousElementSibling.value) >>
this.nextSibling;">click to retrieve a compound</button><div></div></div>
```

Note: The browser recreates a valid HTML document inserting the content above.

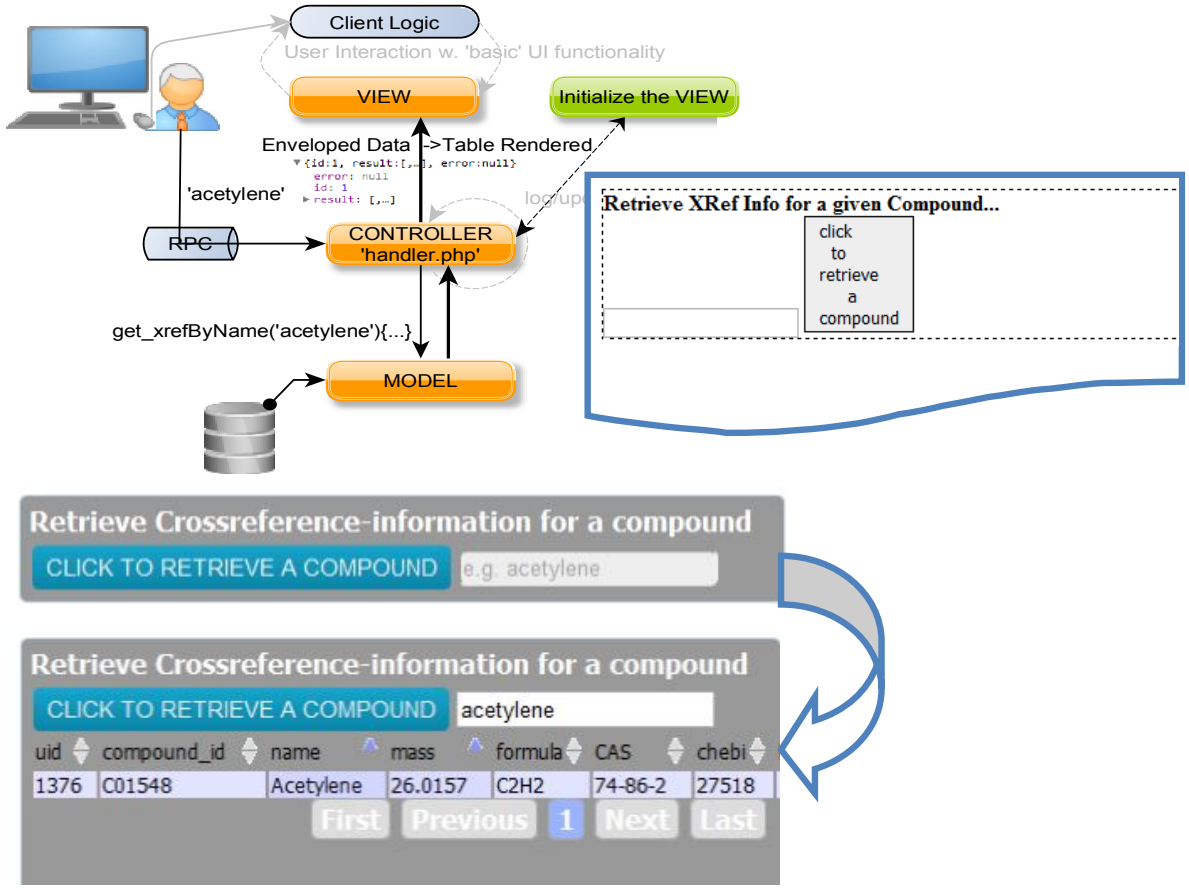


Fig 22 Top: Case study of the MVC Pattern for the exemplary 'chemical-retrieval' web-app: The user enters a compound in the User-interface, and clicks a button, whereupon the *controller* selects a *model* retrieving chemical compound data, to be subsequently returned to the *View* and rendered.
Inlay: The exemplary web-application, inside a WYSIWYG HTML Editor - no HTML knowledge is required
Bottom: A Web-application resulting out of three lines of markup-code and an applied cascading stylesheet (CSS); With one additional line used for inclusion of JQueryTable, the table results are rendered interactively (shown). In this instance, *Datatables*[25] is used for advanced interactive table-functionality

[25] http://datatables.net/

The functional web-application results out of three lines of markup-code or 260 characters and may be rewritten to fit inside a twitter-message of less than 160 characters. The application allows entering of a chemical compound's name to retrieve server-aggregated information of a compound's mass, chemical formula and database crossreference information. Shown in Fig 22 is a Toolkit-built application example, which is further elaborated in the Appendix. Hosting and deployment of applications is possible on freely offered LAMP-server systems, with no requirements towards unconventional PHP-modules.

3.5.2.6 The Aggregator as a web-service provider

Owing to its SOA architecture, the Aggregator is primarily an assembly of exposed web-services which consume, serve or process data related to plant metabolism. Additionally all data queries are cached on the server (and client, if preset) thus eliminating data roundtrips to third party service-providers and speeding up the service response-times.
Additionally caching eases (remote) debugging and grants detailed user-statistics of highly accessed Aggregator information or features to adapt the user interface to the prevailing usage-scenarios, which can be less clear-cut for Metabolomics than for other Omics disciplines (See 1.3, 3.4). Therefore service- availability can be guaranteed even if a service provider is momentarily inaccessible.

Remote SOAP functionality provided by other bioinformatics groups can be directly accessed by first loading the corresponding WSDL resource, and remapping the Client's function-list, as follows:

```
rpc.putWsdlOnServer("http://biomoby.org/services/wsdl/ipb-
halle.de/MetFrag_Query", 'metfrag')
rpc._loadwsdl.push("metfrag")
rpc.Remap()
rpc.metfrag_MetFrag_Query._h
Help for the remote method "metfrag_MetFrag_Query":
#List of parameters:
      @param#1: $data {string} optional:false
#Return value-type of the function:
      @return:string
rpc.metfrag_MetFrag_Query._dbg
Debug: No Trace is set. Log Entries found: #0
rpc.metfrag_MetFrag_Query._mode = rpc.MODE.UNCACHED | rpc.MODE.SOCKET
128
```

In practice, the function *MetFrag_Query* may be 'overwritten' on-the-fly at the server, to mix-in or process additional data or to output additional debugging information.

An example for using the Aggregator web-services is provided in 3.1.2, and the Appendix.

3.5.2.7 Concluding Comparison: Metabolomics vs. Proteomics Aggregator

Whilst the differences of Proteomics versus Metabolomics have been discerned at various points throughout this thesis (1.3, 1.5, 3.5.0), the architectural disparities between two existing aggregator solutions, for two quite different Omics fields shall be briefly outlined in this sub-chapter.
Unlike the MASC-Proteomics Gator (MASCP), this aggregator project takes a pure, distributed service-oriented architecture approach. The need to build upon a different infrastructure is reflected in the greater challenge of facing the data-integration task for a wide-scope of interdisciplinary metabolic-data. Arguably, Metabolomics may feature the most open-ended data analysis of the three major Omics disciplines (R L. Last, 2007),adjacent to genomics and proteomics.
A protein's linear sequence remains widely invariant, lending itself well to a client-side approach of visualizing abstracted data along a protein's primary sequence. This approach is taken by MASCP. In principle Client Side Aggregation (CSA) can yield faster response times as data is directly accessed from the data provider.
CSA was not deemed suitable for the potential data amount that a future-proof Metabolomics portal should handle (See 4.3).
The uncertainty towards presentation, data-integration and future extensibility of a Metabolomics portal was met by using the current state-of-the-art concept of distributed-SOA (3.5.2.1). In contrast to consuming web-services for data-aggregation and subsequent direct presentation, this aggregator primarily serves to provide web-services along with a web-platform built on top of the existing service-base. Delivering and maintaining performance is of key significance for future extensibility. This is facilitated through careful design of data-schemas, data queries, caching as well as through indirect load distribution. Service providers such as cost-efficient Cloud providers which host the SOA-based aggregator, offer additional margin towards future implementations of data and processing intensive tasks, without the immediate need for developers providing code optimization services.

4. Presentation and User Interface

This chapter introduces and analyzes important User Interface (UI) aspects of the aggregator portal.

4.1 Introduction

The user interface of a web-application portal manages the user interactions and presentation of the *View*, in the *MVC*-model (2.3.1), and is also called the *frontend*. The frontend forwards its resulting data requests to server-side components, referred to as the *backend* which implements most of the so called *business logic*. These are functions and algorithms that handle the information exchange between the data and the user-interface (UI). It is the asynchronous communication between frontend and backend, along with client functions to manage the UI and present the data, which make a seamless web-application user experience possible.

4.1 Objectives

An application's UI provides the interaction between the user and the computer. As the UI affects the amount of effort the user must expend to perform a given task, it has a big impact of an application's usability (V Silva, 2009). Significantly decreasing user-time expenditure for performing internet resource-intensive tasks is paramount to the Aggregator's concept and should thus be reflected in the UI decision and improvement process during development.
The aim of a seamless user experience is made possible by adhering to a set of principles which are as follows, a) fast initial loading times, b) no page-refreshes, c) on demand-content loading as well as clearance of obsolete content, d) explicit avoidance of Browser plugins with Java being particularly disruptive in terms of loading times, e) cross-browser support and a f) feature richness which may challenge native applications.

4.2 Summarization of the UI Concept

The user interface of the aggregator portal revolves around three notions. Firstly, the concept of *portal entry* focuses on the UI elements which the user will notice at first sight. Portal entry encompasses the initial time it takes the portal to display. Secondly, a *user-centric* area located at the top, and thirdly, a *result-centric* area located under-

neath each of which is assigned a different color-theme (Fig 23). As a web-application the portal site should never reload.
Functionality, secondary to searching, should only gradually be revealed as the user explores the UI. To this end, a tabbed-UI is well suited. To reduce resource-load on the server and client-side, content is primarily loaded on-demand. A specific remote function allows to asynchronously load content in the tabs at activation. The toolkit client-function '*onvisible*' allows event-driven loading of resources, such as images, upon the UI rendering particular UI elements, with a slight 'consideration period' to avoid unnecessary content-loading during quick user-scrolling. For instance, dynamically displaying a list of hundreds of chemical structures, generated on-the-fly out of SMILES-codes, may become infeasible otherwise. *oninvisible* allows clearing content from the Browser-DOM after a certain timeout period. During the design of complex web-application, more concern has to be taken towards efficient resource-usage, as opposed to the classical design of native applications.

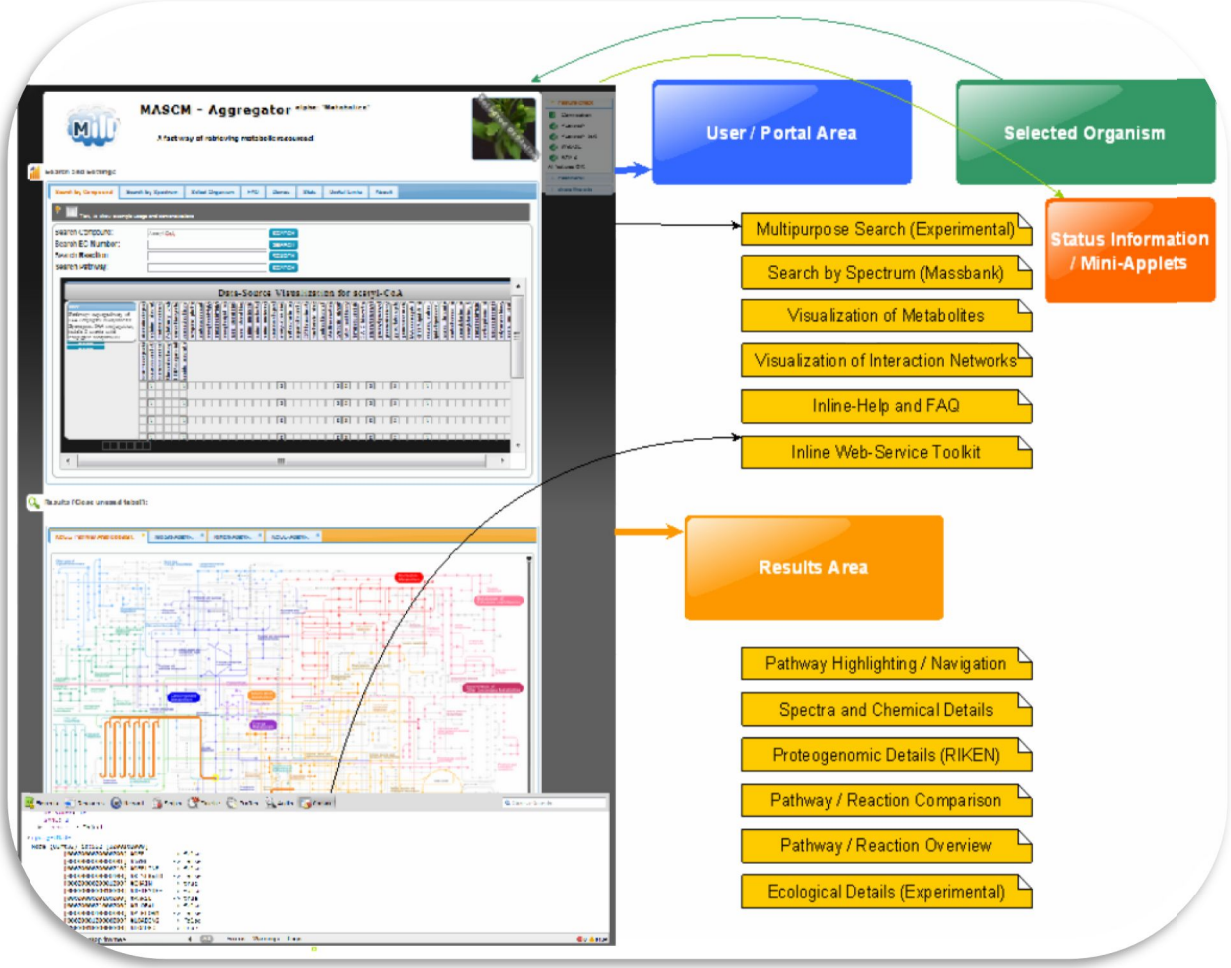


Fig 23 Showing the overall concept of the Aggregator Portal design: Color themes and a top user-area followed by a result-area underneath, are central to the portal's design aspects. The user instantly recognize the search area at first sight, and can familiarize with the search options. Less important functionality is gradually unraveled as the user explores the various tabs and applets (top-right corner).
Each area is assigned a protruding earmark to its left, which can be clicked to expand or contract the corresponding 'area', and to overall furnish a design with vertical and horizontal content extensibility.

4.3 The User Interface (UI) and web-application aspects

When developing native UI-based applications, resource-usage is generally much lower and data throughput much higher. The goal of fast loading times of the Aggregator portal was accomplished by initially displaying and loading few resources, and generally loading resources on-demand. The loading-size footprint is about 0,25MB at the time the client Browser finished loading the portal for the first time, whilst requiring about 10 resources to be loaded via HTTP. To put this into perspective, the well-implemented scientific application of the PhosPhAT[26]-Database (Heazlewood JL, 2008) has a total loading footprint of 0,5MB. The debug version of the portal-page, used during development, is tenfold with a footprint of about 2,5MB. To reduce the number of HTTP roundtrips (request + response), the productive aggregator version summarizes scripts into a single file which is subsequently *minified* through an external application, and images are summarized into image-sprites to be displayed via the *background-position* attribute defined in cascading stylesheets.
The pioneering effort of modern scientific web-application development may require different development directives. For instance deleting DOM-nodes is important, yet it is the Browser to detect unreferenced objects in order to clear them from memory (*garbage collection)*. This process entails a backlog at the cost of memory efficiency. This aspect, when combined with the issue of complex web application-UIs being computing and memory demanding, can result in unexpectedly high memory consumption, leading to caching-issues for users with ceiling memory consumption, and subsequently to an unresponsive operating system, as a worst case scenario.

4.4 The user area

Several Tabs in the User Area (Fig 23) provide a *multipurpose search* which is initially shown (Fig 24) as well as a *spectrum search*, *Statistics*, *Help, FAQ* and Project related information. Fixed at the right top is an *applet-box*, providing information such as the RPC-connection status and the Browser support for required HTML5 features to allow displaying 3D structures via WebGL. Additional applets can easily be added by other developers as part of *extensibility-scripts*. The feedback applet, besides providing a quick means for users to comment or report bugs, also serves as a demonstration-applet for developers.

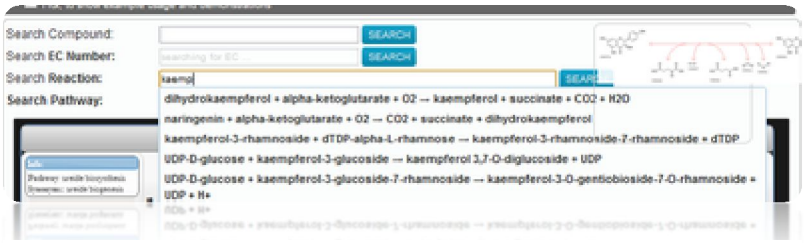


Fig 24 Showng the primary search-fields, with autocomplete-suggestions and a preview, in this case previewing a reaction-equation.

[26] http://phosphat.mpimp-golm.mpg.de/

4.5 Searching metabolism-centric information

The portal provides a simple and quick way to start the search with four input fields, each assisted by an interactive display of possible search-queries (Fig 24). The input masks unfold during searching and provides information regarding the search state, and on-the-fly suggestions, which are entered in the search-mask upon selection. Additionally a search-type dependent preview is provided. When the user clicks outside the search-mask the search is generally automatically initiated.

Following is a list of the search types, possible search patterns and suggested results:

Tab 4.1 Search-Types Search type	Autocomplete Successful	Results/Comment
Compound	*Kaem* *oxy* *kaem;oxy;*	Kaempferol ! multiple compound search is not yet supported
Enzyme	*1.2.* *1.2.2.2* *3.-* *1.2.-*	Oxidoreductases pyruvate dehydrogenase Hydrolases Oxidoreductases acting on the aldehyde or oxo group of donors
Reaction	*->;kaem; UDP;* *UDP;cyan;shis*	Forward reactions containing an UDP compound and a kämpferol derivative 1 reaction containing, UDP compound, cyaniding and shisonin
pathway	*Lipi* *glyco*	4 results for lipid biosynth. 2 results for glycolipids

After the search process is completed, various tabs are loaded in the results-area which correspond to the search-type and selected user-layout. Results can be closed by clicking on the (x) symbol at the top right of each tab, thus freeing Browser resources.

4.5.1 Search Types

A search can be of the type *Keyword, ATG Gene Identifier, Protein (UniprotID), Spectra, Compound*, *EC Number*, *Reaction* or *Pathway* and are autosuggest-assisted as soon as the user enters more than two characters of a search-term. The number of suggestions

is generally set to be limited to a maximum of 10 suggested results.

4.5.2 EC-Number Search

When **searching** for **EC numbers**, the EC class description is provided at the top followed by compounds found within in the searched EC class (Fig 25). Clicking on a compound will place the term in the compound search-field above.

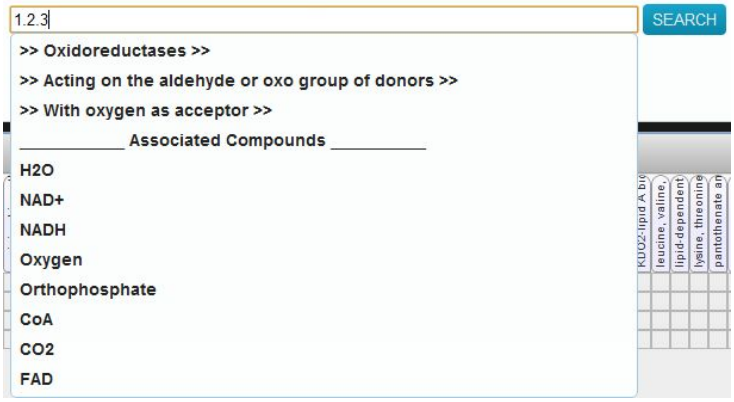


Fig 25 EC number search: EC-class description and corresponding compounds associated with the enzyme/reaction mechanism are provided

4.5.3 Reaction Search

Searching for **reactions** is manifold. An open-ended search possibility presents itself by entering a discrete number of reaction-energy contributing species ("energy currency"), such as for instance searching for reactions containing '*6 NADP+*', '*5 H+*', '*6 NADH*', '*3 O2*' or '*2 ATP*'.
Reaction directionality is added by entering arrows such as '`-->`' (forward), '`<--`' (backward) or '`<-->`' for reversible reactions. The search for '`xylose <-->`' will turn up undirected reactions with xylose as reactant. In a possible usage-scenario, a user could be quickly ensured that no reactions are present for Arabidopsis th. with xylose as a breakdown product, by searching for '`xylose <--`'. A web-portal with instant loading-times may be the fastest way of retrieving such information.
Additionally, reactions may be searched by entering up to four reactants or products, separated by a semi-colon ';'. e.g. *'L-tryptophan; pyruvate'* (See Tab 4.1)

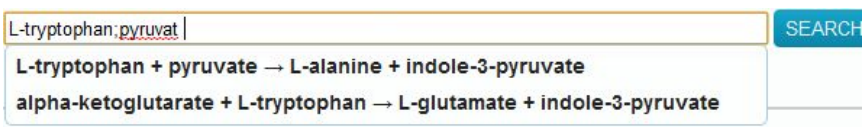


Fig 26 – Demonstrating the multiple reactant search mode-option, when searching for Reactions.

Searching for the term *'unkown'* will list reactions for which no enzyme has yet been determined based on the *AraCyc v8* dataset.
Upon selecting the desired reaction from the autocomplete-list, a new result tab details reaction information compiled from *KEGG*, *PMN* and *BRENDA*. All com-

pounds/reactants within the selected reaction are treated as a set of compounds, which are queried and visualized. For instance, the reaction *UDP-D-glucose + D-fructose ↔ sucrose + UDP*, will result in an attempt to visualize the set of compounds comprising *UDP-D-glucose*, *D-fructose*, *sucrose* and *UDP (Fig 27)*. As part of future work, clicking on the compound-button may provide a menu option to add the top 5 corresponding substructures of the clicked compound to the visualization-grid. As such, for the previous example (Fig 27) it would be sensible to include *D-glucose.*

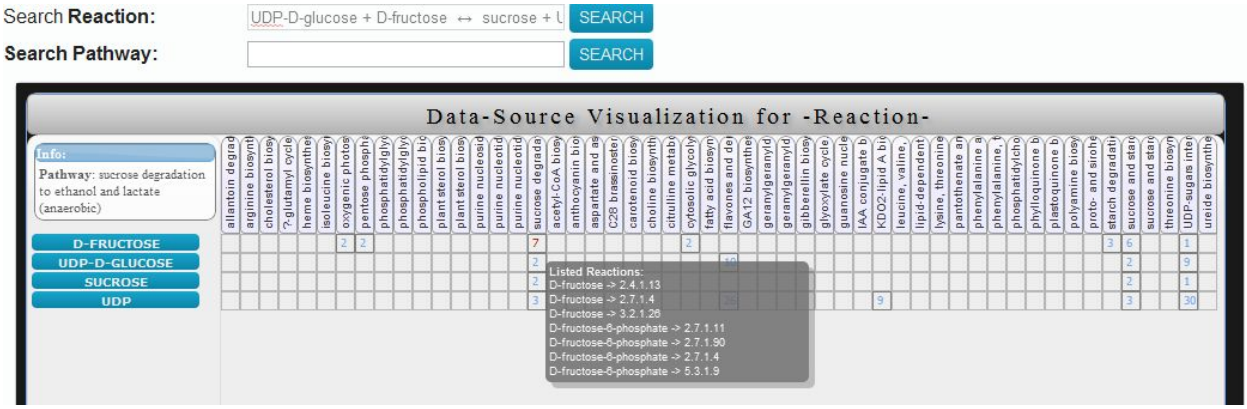


Fig 27 Visualization for the Reaction UDP-D-glucose + D-fructose ↔ sucrose + UDP (see Text). The initial search term was 'sucr'. The highest occurrence of UDP is found in the *UDP-sugar interconversion* superpathway. As work in progress, isometric bars may express the number of occurrence as a heightmap.

4.5.4 Pathway Search

Searching for a **pathway** is performed by entering the pathway name, and selecting any of the returned results of the autocomplete list, to bring up a detailed result-tab.

4.5.5 Keyword, Gene, Protein Search

Universal search terms allow searching by Uniprot IDs or A. th. Gene Identifiers (ATG) in any of the aforementioned search boxes (e.g. searching for '*AT1G23'* or '*Q9LNJ4')*. Keyword searches are initiated by double quoting the search-term e.g. *"vacuole".* At the moment the search does not discriminate on where of the four search-fields the universal search-term was entered, thus yielding the same auto-suggested results.

4.5.6 Spectrum Search

Searching by a Mass-**Spectrum** can be performed by a drag-and-drop operation of a *msp*, *msl*, *mgf* -file or a similar text-based peak-list file. The user can conveniently utilize the *Massbank* search feature from within the aggregator portal, and explore the results further via the portal-provided data aggregation functions. By selecting a chemical compound from the returned results-list, a new result-tab with details is added in the results-area of the UI.
Alternatively to dragging and dropping spectra-files, conventional uploaded via a file-input element is possible (Fig 28). The spectra is rendered interactively in a HTML can-

vas-element, with the option of zooming by selecting an canvas-area of peaks with the mouse. The best matching compound can be directly explored as a 3D model pop-in and details are provided in the results area.

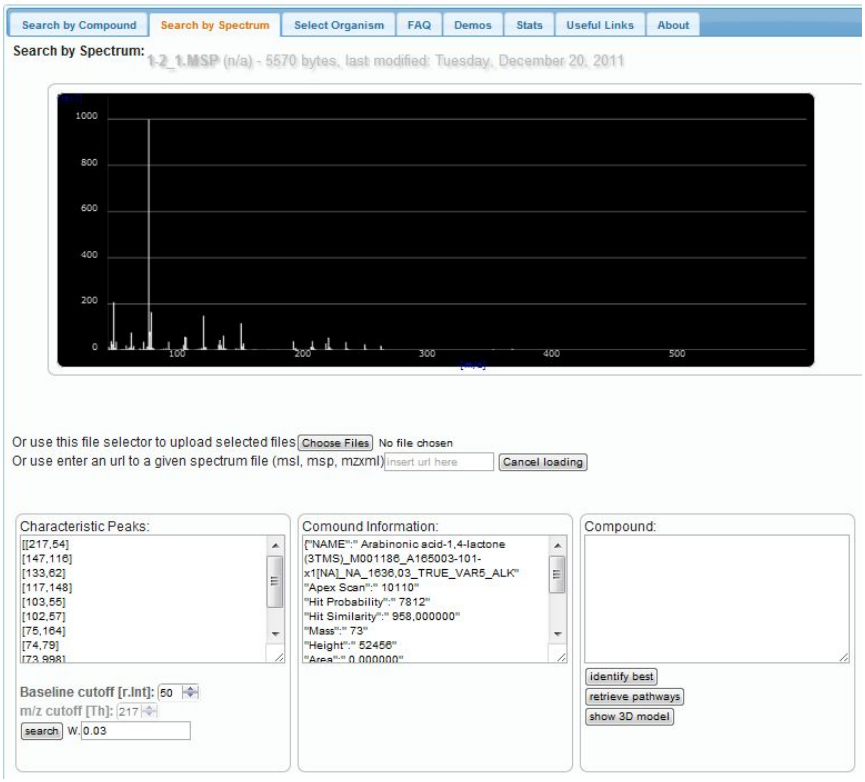


Fig 28 Dragging and dropping a spectrum, putatively identified via LECO ChromTOF as Uridine 5'-diphospho-N-acetylglucosamine from a previous experimental run. The peaklist was exported as an msp-file via LECO ChromaTOF. Note: Experimental UI; subject to change.

Example:

The spectra of a deconvoluted msp-file (see Appendix) from an previously performed experimental run, and putatively identified as Uridine 5'-diphospho-N-acetylglucosamine, yields the following massbank results:

Tab 4.5 – Results from a msp-file Spectra Search / top 10 results only

score	Compound-name	formula	mass	MbID
0.855854805868	Uridine 5'-diphospho-N-acetylglucosamine; GC-EI-TOF; MS; n TMS; BP:73	C17H27N3O17P2	607.08157	PR010241
0.842348814890	L-Ascorbic acid; GC-EI-TOF; MS; 4 TMS; BP:73	C6H8O6	176.03209	PR010206
0.836871445517	L-Ascorbic acid; GC-EI-TOF; MS; 4 TMS; BP:73	C6H8O6	176.03209	PR010208
0.833422529240	L-Ascorbic acid; GC-EI-TOF; MS; 4 TMS; BP:73	C6H8O6	176.03209	PR010207
0.778210309721	L-Iditol; GC-EI-TOF; MS; n TMS; BP:73	C6H14O6	182.07904	PR010085
0.767568955832	Ribitol; GC-EI-TOF; MS; 5 TMS; BP:73	C5H12O5	152.06847	PR010135
0.765011819877	D-(-)-Fructose; GC-EI-TOF; MS; 5 TMS; 1 MEOX; BP:73	C6H12O6	180.06339	PR010041
0.759831611656	Uridine; GC-EI-TOF; MS; 3 TMS; BP:73	C9H12N2O6	244.06954	PR010029
0.728153658181	Ribitol; GC-EI-TOF; MS; 5 TMS; BP:73	C5H12O5	152.06847	PR010134
0.687301773496	D-(-)-Fructose; GC-EI-TOF; MS; 5 TMS; 1 MEOX; BP:73	C6H12O6	180.06339	PR010042

Spectra are automatically uploaded to the server and stored for a predefined period of time. In case of the user being logged in, the User can choose from a list of spectra uploaded in the past, which are present on the web-server. The process of the user-login does not require the user to reveal sensitive information or user–credentials such as a password and username, but is performed via the OAuth v2.0 protocol and a trusted third party authenticator such as Google, Yahoo, Facebook etc. User-Login (Fig 29) is crucial for usage of work-in-progress social and community aspects of the portal, such as spectra sharing, voting on literature pertaining to a given search-result and spectra annotation.

Fig 29 Login opens a sign-in box using Janrain-engine and the OAuth protocol.

4.5.6 Compound Search

When a **search** for a chemical **compound** is performed, the corresponding compound is highlighted in the Pathway-Reference map of Arabidopsis thaliana. Upon clicking on the compound additional information is retrieved via KEGG. A particular enzyme can be highlighted in the map to reveal all other locations within the pathway map. Likewise a search in which an enzyme/reaction is involved will highlight the corresponding pathway-sections (Fig 30). Several result-tabs are opened for the compound, depending on the selected user-layout. Spectra, Physical, Chemical, Medical and Biomarker information is provided (see 2.5) via several tabs in the results-area.
Additionally the compound is added to the Compound/Pathway Visualization (Fig 30)

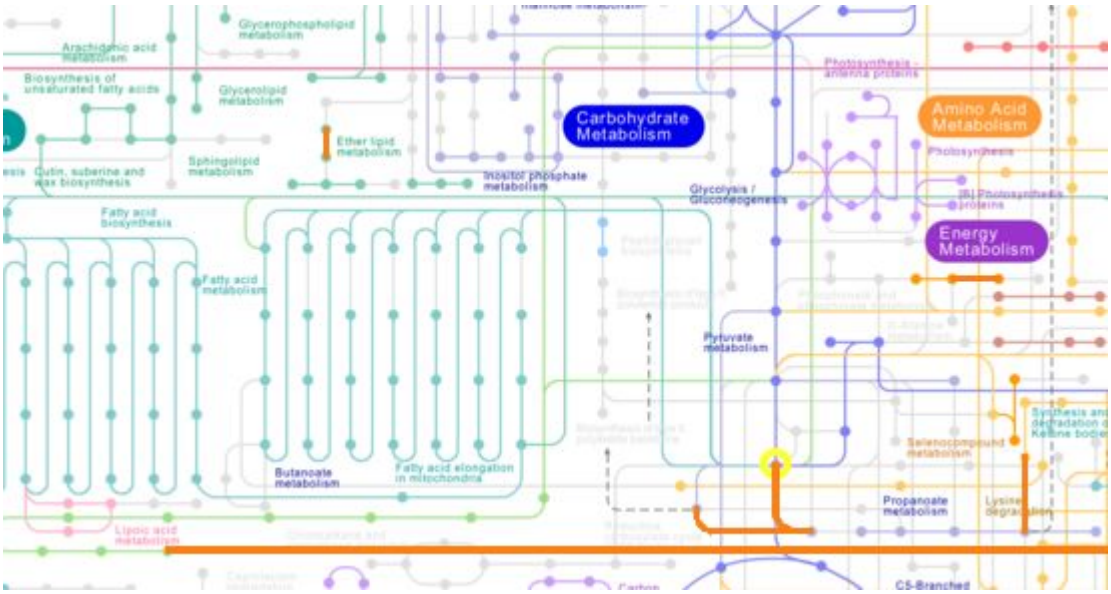


Fig 30 Acetyl-Co A was searched and is highlighted. AAPT1, HMG1,HMG2, ALDH (ALDH6B2) is selected, and can be further explored via the portal's functions. The Reference map image is provided by KEGG, and can be zoomed in via the mousewheel.

If another organism is selected its corresponding reference pathway map is loaded, shown in Figure 30 for Arabidopsis thaliana. Time constraints and data curation effort currently limit the organism to A.th. As an experimental feature, other organisms can be selected (Fig 31).

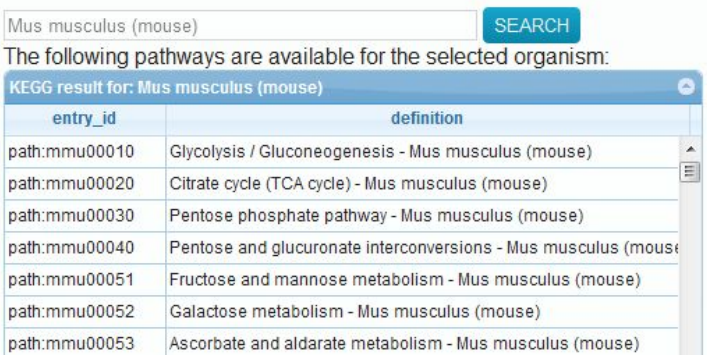


Fig 31 Left: Selecting a different organisms (right figure) is possible as an experimental feature, and changes the organism indicator at the top (left figure).
Right: Double clicking on the pathway will bring up a pathway-overview via KEGG

Shown in Fig 32 is a list of Spectra loaded in the Spectra-tab upon searching for Acetyl-CoA. Spectra are retrieved from Massbank. As part of future work, a comparison feature may allow overlaying spectra and selecting matching peaks, which are entered in a textbox for user-selection and download or to yield a new spectra search based upon the user selected peaks.

Fig 33 shows the Reactions/KEGG tab when performing a compound-search. By clicking on the blue-title-bars, the result-inlays are expanded and loaded on-demand from KEGG.

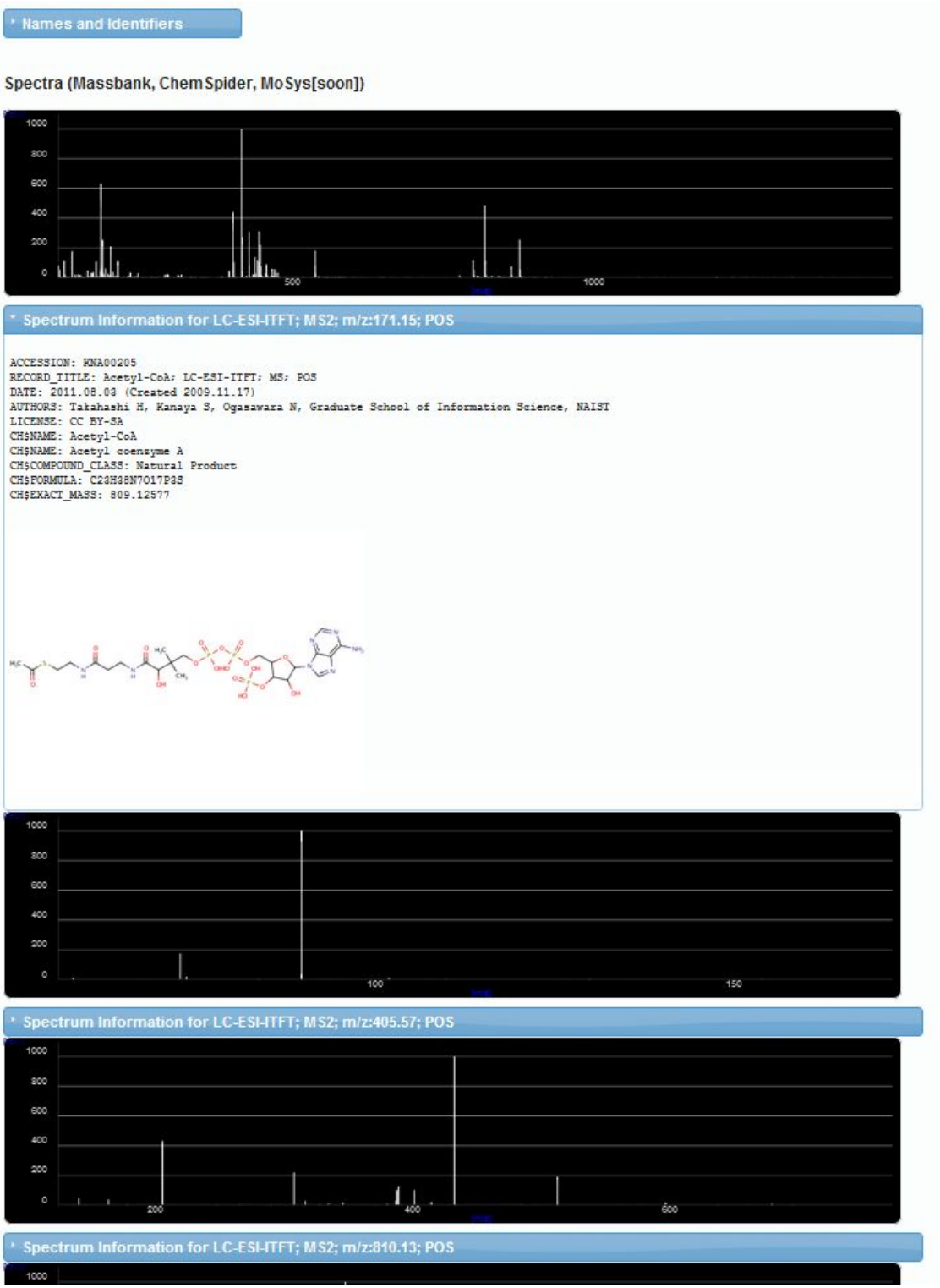


Fig 31 Spectra Overview in the Spectra-result tab when performing a compound-search for 'Acetyl-CoA'. Spectra shown are courtesy of massbank.jp. 2D structure views are generated automatically from SMILES strings, via a call to a web-service. Clicking on the blue title-bars shows additional information on-demand. Underneath follows a table of ProMex Spectra and Plantmetabolomics Spectra.

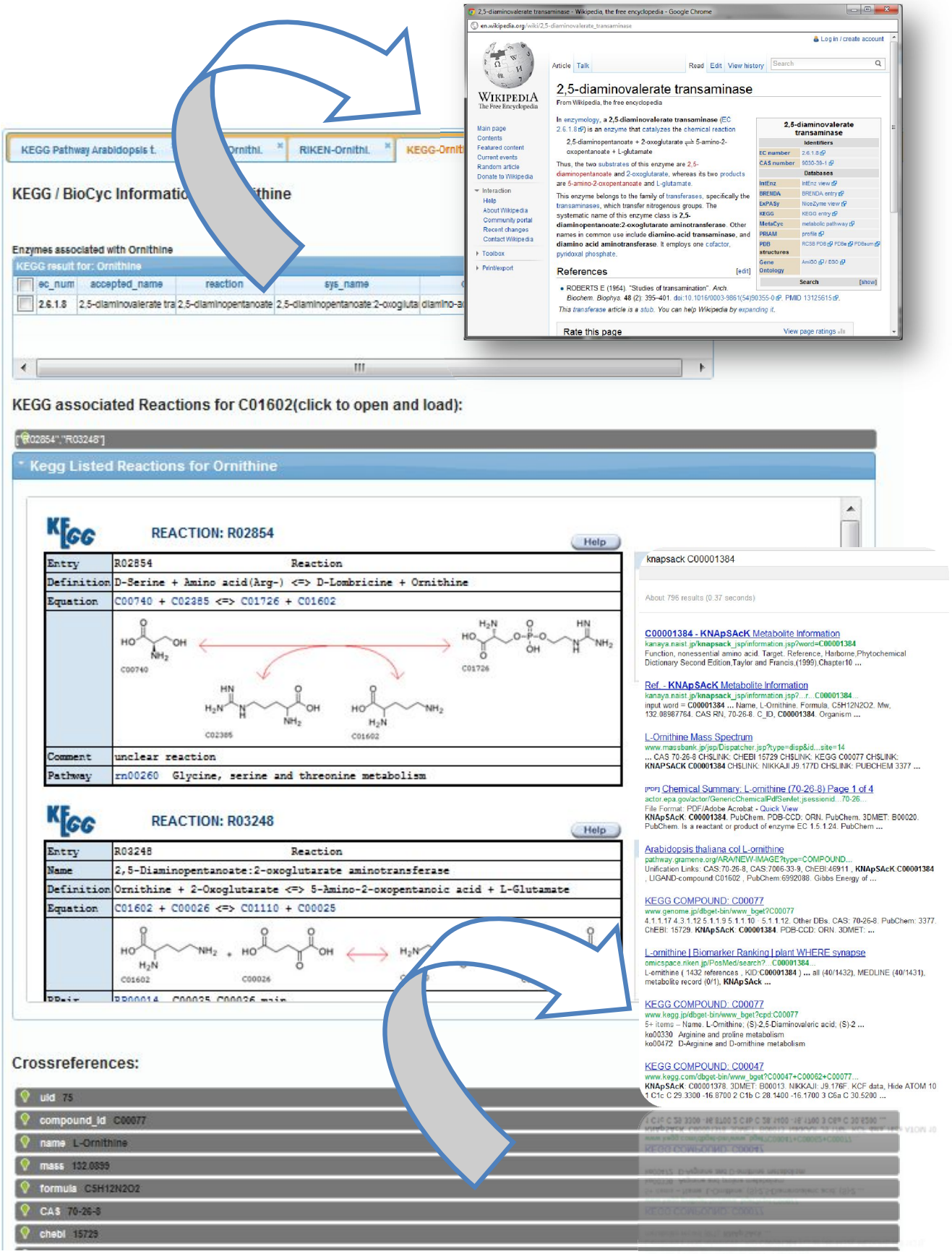
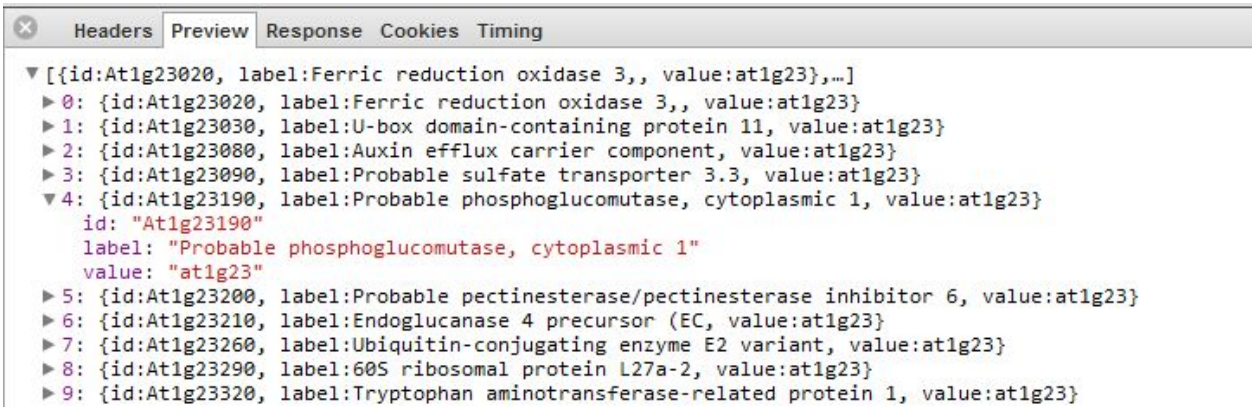


Fig 32 Showing the result tab with pathway and reaction information for 'Ornithine'. Links to BRENDA, EXPASY, IUBMB, KEGG, Knapsack, Chebi etc are provided. Links are often mediated via google, to furnish a more comprehensive overview of a particularly resource. Inlay: google results for 'knapsack C00001384' (Bottom inset) Results are loaded on demand upon expanding the blue title bars.
Top-Inset: Double clicking on a table-entry will provide comprehensive enzyme information via Wikipedia.

4.5.6 Search Providers as RESTful web-services

Searches are facilitated by a *search-provider* web-service, which returns JSON encapsulated results via a HTTP response after a HTTP GET or POST request. In the request's url parameters, *db* determines the search-provider and *term* the search-term which is automatically stripped for whitespace-characters on the server-script. Additionally the search-provider is exposed as a RESTful service outputting XML or JSONP data, to allow cross-site-scripting (XSS) implementations by other developers.
In the following example of the resource `/gator/search/ec/1.2.3.-/JSONP/myfunc` , *myfunc* denotes a user-specified JSONP callback function-name, which is set to *'searchmgate'* if none is provided and *ec* selects the search-provider domain. Other REST-like services include a service for SMILES conversion with content-caching e.g. cached/img/smiles/C1(C(=CC(=C2(C=1OC(CC2=O)C3(C=C(C(=C(C=3)O)O)O)))O)[O-])

A simple GET-request to the resource gator/search.php?db=ec&term=AT1G23
will yield the following JSON data:



A server-side declaration of a new search domain may look as follows in PHP-syntax:

```
    case 'searchproviderName':
      $items = fetchArray(
        "SELECT * FROM `organisms` WHERE definition LIKE '%$q%' LIMIT 0,30",
        'definition,entry_id'
      );
      break;
```

Distributed aspects apply to individual search-domains. A complex search-scenario, may requires multidimensional-drilldown with subsequent aggregation, which as an atomic search provider-domain may be delegated to an external script or background process or to another service or function domain communicating via an RPC (see 3.5.2.1, 3.5.2.2).

5. Results, Discussion and Outlook

This chapter concludes this work with a project-recapitulation, results and discussions, and a future outlook.

5.1 Recapitulation of the Project Achievements

The Aggregator application is designed to visualize Data related to metabolism or Metabolomics in a simple format. One aim is the quick exploration of a great number of available resources by entering one of seven search-types, encompassing *spectra*, *compounds*, *genes*, *proteins,* r*eactions*, *pathways* or *"text"* (see 1.1, 4.5)

Requiring only a short learning period, the consistent interface allows interactive exploration of compounded metabolic data to aid in furnishing a holistic picture for scientists. The Aggregator, as a web-portal *frontend*, poses only a user-interface, whose design considerations are directed towards consistent presentation as a tradeoff between user-interface visuals and the challenge of displaying highly heterogeneous data. (See 2.5.3, 3.1, 4.2)
As a whole, the User interface of the Aggregator attempts to be kept tidy, promoting a consistent-design and color-themes to discern key-areas, whilst maximizing information content by being vertically as well as horizontally extendable (See 4.3, 4.4).
Any other user-interface may be linked to the *aggregator backend* or web-service functionality, such as the service-centric workflow studio *Taverna* (D. Hull, 2006), which is more amenable to bioinformatics workflows (See 3.5.2.6, Appendix).

At the moment, data presentation is restricted to Arabidopsis thaliana, and data federation for even a single organism poses to be difficult. As part of future work, the Aggregator could be extended to encompass other model organisms. The selected Organism is visually indicated in the portal at the top-right corner, with suitable images automatically retrieved via Google Images. Automation is a key aspect to the Aggregator for all tasks where a curator is expendable (See 3.3, 2.4.5.2, 4.5.6, 5.4).

Additionally the provided Aggregator toolkit may allow to quickly transition from the conceptual stage of a scientific web-application project to a functional draft, which can be put in a productive environment with little additional effort (See 3.5.2.1, 3.5.2.5).

5.2 Results

5.2.1 Accomplished

Within the scope of the project three core elements are provisioned, comprising a toolkit, a broad range of web-services and a web-application portal which utilizes a subset of these web-services. The aggregator project allows rapid building of scientific web tools, through its underlying small toolkit. Utilizing the toolkit sets the stage for any project as being highly modular and scalable through a distributed SOA architecture. Choosing this architectural option early on proved to be a rewarding decision, as loosely coupled modules require less 'visceral knowledge' about any individual function by the developer. The level of knowledge-detail required to develop and maintain a given project may decide its long term future.
The decision of creating a simple toolkit was influenced by starting out with a comprehensive web-toolkit (*Dojo*), which did not cope well with the demands put on complex scientific web-applications. Web application development offers the developer a limited budget in terms of loading-times and resource footprint. Unlike native application-development, network latencies and bandwidth matter in addition to web-applications incurring an order of magnitude greater memory consumption and speed penalties for rendering the user-interface essentially out of a DOM-based web-document. The change in direction of the project and rewriting the dynamic UI from scratch accomplished a small footprint with almost instantaneous loading even under low-bandwidth high-latency conditions. Thus sufficient leeway towards future usage of scientific visualization- and other web-toolkits is afforded (See 4.3).
An important facet to the aggregator was the idea of providing an embeddable, integrative visualization web-applet, emphasized in chapter 5.2.3, which accomplishes to furnish a quick and useful overview between the functional relationship of pathways and metabolites. By looking at the reactants of an enzyme, and the pathways the compounds distribute to, as well as the relationships between these pathways yet similar compounds, hypothesis creation towards biomarkers may be facilitated – a key impetus for current design decisions and usage scenario directions.
New sources and aggregation functionality is often incorporated in a manner of minutes. This claim can be summarized as being based on a) 'convenience' RPC-functions for rule-based content extraction of remote structured-documents, b) the option of automatic rendition of server-responses in an interactive table-format (See 3.5.2.5, Fig 22), c) direct incorporation of an entry summary or details view of other projects by passing the corresponding accession parameter and finally d) the option of connecting the R-program environment to web-visualization toolkits via an RPC. Additional convenience functions allow directly invoking a 'moderate subset' of SQL-database queries

at the Client side, with results subsequently automatically presented as an interactive table or loaded into an interactive visualization.
Lastly, a list of bioinformatics services and data has been compiled over several months. Almost ten data providers have been incorporated, most important of which are KEGG, TAIR and Aracyc.

5.2.2. Missing

One of the initial aims of the web-portal was to provide users with multiple integrative visualizations, yet much of the visualization web-applets are still missing, despite the presence of technical layer for dynamic visualization and basic backend-aggregation functions. Design aspects (4.1, 4.3), along with improvements and revisions of existing concepts, contribute towards a greater delay of any particular visualization-applet than would be the case for a simple proof-of-principle user-interface draft. UI design concerns towards Metabolomics are generally challenging as some usability characteristics of the web-portal are still unclear (3.4).
Additionally no attempt has yet been made to integrate the vast number of web-resources compiled over several months into the portal. Preceding this work is the desire of semantically annotating and describing the entries within this metabolism-centric list of resources, as well as rating a resource's significance for this project.

Another characteristic which is lacking, are well delineated, tidy user-workflow scenarios, which would inherently present development-opportunities regarding feature improvement and presentation. By keeping track of user-workflows such as the chronology and nesting of RPC function invocations, new aggregation functions may eventually be derived on the basis of wrapping and moving frequently used workflow-choreographies to the server-side whilst phasing-out or externalizing infrequently used aggregation functionality.

When adhering to the outlined UI aspects (4.1, 4.2), the user-area should command the result area, rather than a mutual interaction wherein the user browses results in a manner that feeds-back search-suggestions into the search-fields located in the user-area, thus violating a logical UI workflow hierarchy. This may be fixed by offering search options directly at the location of the mouse-click within the results-area.
Whilst concerns of this nature may sound petty to many scientists, the user-workflow of the UI is similarly important as the loading-time of the web-portal. When building content tools *it should be clear* that two preconditions apply, *domain expertise* and an *in-depth understanding of the real-life workflow* (Elsevier, 2012)

Further considerations of this nature demand narrowing down a set of key UI-design rules, which are missing at the present.

Missing too, are abstracted data-workflows for each aggregation function which would delineate sources and targets and how data is mixed and enriched in between. These workflows can be openly shared via *myExperiment* (JISC, 2010)[27], and lend themselves well to presentation. Strict adherences to several such workflow outlines are likely to make a significant impact on the usability, and thus long-term success of the aggregation portal in addition to aiding the API-documentation.

The aggregator toolkit provides several ETL scripts to aid the process of federating widely static or infrequently updated resources into a single data object. However, the scripts are not applicable to general purpose scenarios. ETL processes are still manually-driven laborious tasks, which require meticulous devising of data-schemes and are, within the scope of the portal, generally lacking behind the incorporation of external web-service based data providers.

As treated in chapter 3.4, precise usage scenarios of the portal are missing, but with a growing number of features, along with user-data, this situation is likely to improve.

Finally, from a project-point of view, automated testing-scripts are desired, as well as project funding, additional developers, more community features for the portal, and a detailed description of data-sources similar to the XEWA (XML-Enabled Wide Area Searches (Critchlow, 2001) or Bio2RDF project (B François, 2008).
The project has yet to face tasks of code-cleanup, restructuring of the monolithic source-files, as well as solidifications towards security-concerns, and incorporation of compiled data-sources.

5.2.3. Presentation

Three novel visualization techniques were devised which are presented below. A pathway-compound *visualization web-applet* provides an web-embeddable, quick overview of the relationship between reactions or *individual compounds* and pathway-clusters or *superpathways*, as shown in Fig 33.

[27] http://www.myexperiment.org/

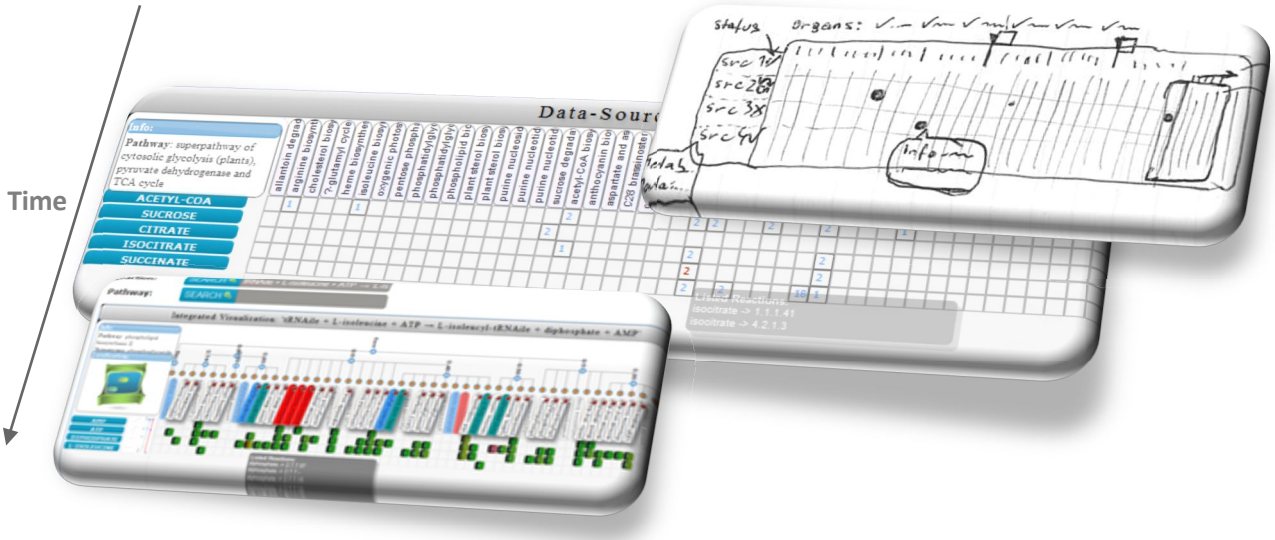


Fig 33 Schematically showing the evolution of the visualization overview. Superpathways/pathway-clusters are listed at the top from left to right. The number in the visualization matrix indicates how often a particular compound is present within the sum of the reactions of a given pathway-cluster. To the left are buttons which bring up a menu for the compound, labeled by the compound-name: Shown are Acetyl-CoA, Sucrose, Citrate, Isocitrate, Succinate. Above the vertical listing of superpathways will be a dendrogram generated upon user-defined distance measurement criteria. Hovering the mouse over a particular field shows additional information such as reactions for the compound within the superpathway, and clicking on a particular grid-field allows further investigation of the reaction through new tabs opened in the results-area. Searching for another compound adds the result to the visualization-grid. Clicking on the compound-labeled button may allow for instance loading the ten most similar metabolites of A.th.
Additional information could be overlayed via isometric bars, and different coloring. The visualization can be popped out or embedded and shared, in which case a url shortening service (goo.gl) is used. As a web-applet, in contrast to a static image, this method ensures always up-to-date information.

A quick summarization of data sources is provided by a bar chart (Fig 34), and experimentally incorporated. The advantage, towards this kind of visualization lies in quickly conveying the overall relevance of a particular compound in terms of the available number of data sources in total and within a particular databases, for each compound of a given pathway.

Fig 34 Showing the number of total compounds, and how often the compound is present in a particular pathway-cluster and in what data sources (dark blue: KEGG, light blue: TAIR); Rendered via Google Chart API

A third key visualization is accomplished via the KEGG reference pathway maps, useful to allow visually navigating and exploring likeness-criteria, such as similar compounds, pathways or sets of pathways and their corresponding enzymes (See 4.5.4) . As KEGG

(2.5.1) increasingly ventures into data and literature mining, a vast amount of aggregated information is directly accessible from within the portal's interactive KEGG-pathway 'explorer'. Moreover, as a central design aspect, adherence to KEGG's color-schemas, for instance *orange* relating to pathways of amino acid biosynthesis or *blue* for energy related pathways, is attempted throughout the portal's design for pathway related information. The KEGG coloring schema is provided in the Appendix.

5.3 Discussion

5.3.1 Challenges

A challenge which could easily be fixed in the scientific community is the unavailability of a centralized repository for database *cross-reference* information. That such a centralized effort is feasible, is confirmed in the form of the Digital object identifier (DOI) with more than 50 million doi's assigned till 2011 (DOI, 2011). One reason for this situation may be, that initially in science greater emphasis was given towards the paper rather than publishing and provisioning of generated data. Indeed, mass spectrometry and Metabolomics is no stranger to the application of *hamburger to cow* algorithms, reversing the situation of traditionally published data wherein later analog information from a publication including structures and molecular spectra are converted back to digital information (Kind T, 2009).
Knowing the *Where* (data-source and location), *What* (data and metainformation) and *How* (crossreference database Identifiers) any elementary aggregator can be built in a very quick manner. The remaining challenge then is *integrating* the data and presenting abstracted data in a novel way that can furnish scientists with visuals that foster new insight.
Another posed challenge lies in the dearth of fertile *visualization* publications that would aid the establishment of *integrative Metabolomics* portals.
The lack of *metadata* for published datasets of Metabolomics experiments results in inferior information besides potentially well devised experiments, although the situation could easily be improved. An example for well annotated metadata is found in the form of the RIKEN datasets (Data-Archive, 2011). Mining of experimental metadata is important for the integration of Metabolomics data (O. Fiehn, 2007).
Metabolomics itself poses great challenges. The non-uniform measurement methods, and much finer physical granularity and timescales differentiate the field from Proteomics and Genomics – two fields based on the identification and characterization of macromolecular species.
The presence, location and abundance of a chemical species can fluctuate on orders of magnitude shorter timescales (Weckwerth W. , 2003). Chemical abundance of a spe-

cies and therefore detection in a metabolomics experiment can depend on the presence or absence of a catalyst and physical conditions such as pH and temperature, as well as on the measurement platform and experimental settings.
To complicate matters, compounds may yet only elicit biological response such as enzymes which facilitate metabolite-degradation at certain concentration thresholds, as a result of biochemical processes such as signaling cascades, leading to fluctuations of metabolites at the milliseconds time-regime (W. Weckwerth, 2005).
Even if all chemical processes were frozen in time for the experimenter to carry out measurements, not-all compounds can be measured in a single experiment and thus with the same measurement-bias or compounds be completely separated and measured individually, owing to widely different physical characteristics of metabolites.
It is thus important and sound to keep a critical stance towards the steady-state compound abundances contained in Metabolomics datasets, and its implications for the emerging field of *integrative Metabolomics,* including intuitively visualizing the uncertainty contained in Metabolomics data and corresponding metadata.

Generally speaking for the field of MS-based Metabolomics, well separated, high resolution MSn-spectra of metabolites for different ion kinetic energies would be desirable, which allow elemental decomposition, once well calibrated (MassWorks, 2010) and isotopically filtered (T Yutaka, 2003), to furnish structure elucidation through incremental algorithmic improvements (T Kind, 2010). Thus data from modern measurement platforms provides an additional benefit.

Web-applications, whilst considered a driving technology for biological sciences, poses challenges in terms of asynchronous data flow and workflow choreographies (5.2.2).

Finally the challenge of explaining a complex tool such as the aggregator web-portal to a potentially large userbase, can be met via subtitled or narrated screencasts, which can easily be put on web-video-platforms such as *youtube* to be subsequently embedded in a web-portal's Help section. Additional help can be provided in the form of a short image-based manual which explains the user-interface with popout comments.

5.3.2 The Aggregator web-application

The Metabolomics aggregator may anchor a unique position in the Arabidopsis community by providing a platform for aggregating and visualizing metabolism-related data around Arabidopsis th. datasets, a toolkit geared for biologists and a broad range of web-services. The underlying SOA architecture is open-ended with the possibility of similar data offerings, eventually serving the *Populus trichocarpa* and *Oryza sativa* community. The implementation of visualizations and data integration requires a high-

ly collaborative endeavor between portal developers and data providers (HJ Joshi, 2011), who frequently already devised means for static visualizations for their data, which can be amenable to enriched and improved presentation methods. Development and integration is arduous, presently limiting the data integration to the model organism *Arabidopis thaliana*, yet team effort, application of new tools and improvements in data-workflows may overcome these developmental hurdles. At times, initial prototyping of interfaces and visualizations did not produce fertile results, because the display of information remained too trivial, too ornery or too crowded and complex. It is at those moments during development where the pioneering status of visualizations for integrative Metabolomics becomes apparent.

Scientists can retrieve up-to-date information at once, and with incremental Aggregator updates may be offered new or improved integrative visualizations with an underlying systems biological viewpoint. Visualizations as web-applets store their internal state on the server and can be embedded for community discussions within any webpage through widely popular cross-site discussion platforms such as *Disqus*[28] - wherein the discussion-platform itself constitutes a 'web-applet' easily attachable to any site.

As a Metabolomics portal the Aggregator offers a platform that attempts to integrate chemical compound, enzyme and therewith protein-, reaction, pathway and literature information. The first step in that undertaking was compiling a list of resources by their relevance towards plant metabolism. Next a platform was conceived and built around a SOA architecture which significantly improved the speed of integrating data-sources. Finally the efficiency of this integrative effort, as it relates to the user-interface experience, will likely improve with increasing user feedback and Portal revisions.

Concluding, to serve the Metabolomics community, compound information puts particular emphasis on the display of spectra. Pathways are put in context of KEGG reference maps and linked with literature and reaction information from AraCyc or PlantCyc. Visual pathways can be explored interactively. Reaction searches allow ruling out certain reactions on-the-fly (4.5.3) and provide means to retrieve information on an entire set of compounds found in the particular reaction. Mass spectral searches allow experimentalists to directly verify a compound and utilize the Aggregator functions. Enzyme information provides reaction and proteomic information with the option of a link-out to the MASCP Gator. A significant aspect, currently lacking in the aggregator prototype, is the focus on metabolic network information and network parameters, despite their significance to systems biology.

[28] disqus.com

Ultimate aim of the project is the provision of a tool that puts metabolic and metabolomic resources in a *functional context*. Such a tool could offer the promise of improved Biomarker discovery (Kell, 2007) and a better understanding of system biological processes.

5.3.4 Integrative Metabolomics

As pointed out by A. Smith et al., Data integration is of primary concern to the field of biology at the moment, facing the widely unresolved challenge of cross-referencing between the genome, transcriptome, proteome and metabolome (A Smith, 2011). A similar plea is made by (E Schwarz, 2012) (See also 3.3). Fig35, from *BIMSB*[29], shows the identified data hierarchy and BIMSB disciplinary groups in the effort of integrative Metabolomics (See 5.3.1).

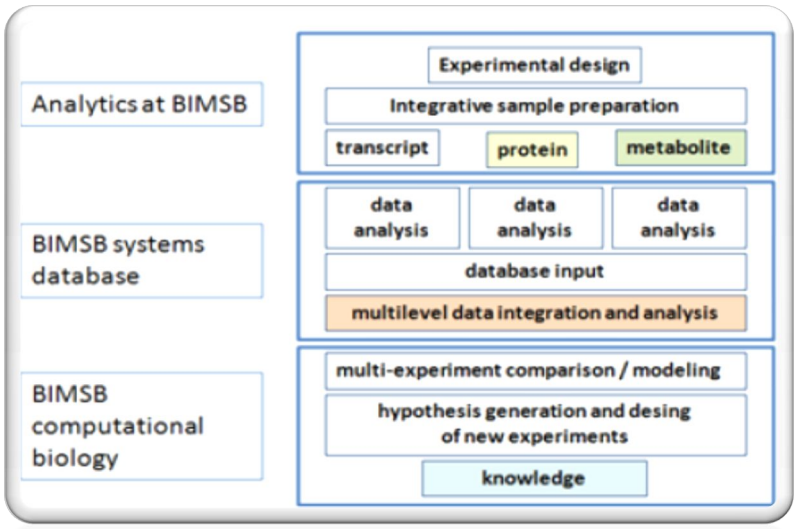


Fig 35 Showing the hierarchy of Integrative Metabolomics

Source: The Berlin Institute for Medical System Biology (BIMSB), by Stefan Kempa

5.3.5 Community Metabolomics

Since the state of a particular visualization is saved and accessible via a url, it can be embedded as a web-applet and, in connection to a discussion system such as *Disqus,* (5.3.2) assist the interpretation of a particular visualizations through a community effort, thus effectively widening the accessible expertise- and interpretation- space. Spectra can be shared, and as part of future work, peaks or a set of peaks annotated. Moreover, literature, as it pertains to a particular search-result, could be rated by the

[29] www.mdc-berlin.de/en/bimsb/index.html Berlin Institute for Medical Systems Biology

user, thus providing a measure on how helpful a particular literature was in the context of a particular compound, reaction et cetera.

A wide range of community aspects could be integrated in the future, to augment the portal with information which would otherwise not be accessible from any given data-source.

5.3.1 Cloud computing in Metabolomis and other Omics sciences

Cloud computing, briefly mentioned in 3.5.2.3, is the evolution of Grid computing, wherein computing resources are provisioned automatically or '*elastically*' , and may be regarded as a black-boxed computing-grid, that offers attractive costs, owing to a competitive market. The cloud emerged from large web companies like *Amazon* who had to invest in massive infrastructure to essentially run one or a few applications (Lew Tucker -CTO Cisco, 2010).
The recent free usage-options, up to a certain resource quota which are offered by major Cloud vendors such as *Amazon* and *Google,* led to a major incentive for developers and therewith a popularity-jump of Cloud computing, as opposed to the preceding gradual adoption seen with grid computing.

For researchers in life sciences, cloud computing offers the benefit of testing computationally demanding algorithms or hypothesis on large data sets with a shallow learning curve.

A cloud computing example in life science is presented with *Schrödinger*, a leading supplier of molecular-simulation and computational-chemistry software to the pharmaceutical industry. Schrödinger recently made its *Glide* docking program available on the Cloud[30]. Glide is used for virtual screening, a process that determines potential drug candidates from a large database of compounds based upon their fit with a given target site.

A major difficulty in Metabolomics is the handling of massive data streams resulting from the experimental measurements, leaving daunting requirements for informatics (M. Milburn, 2006). With a staggering number of datasets amassed over a decade, and increased throughput and resolution of the current generation of Mass-spectrometric platforms, the application of distributed computing, to a potential '*big-data analytics*' issue, becomes evident.

[30] http://www.schrodinger.com/products/14/5/

An important goal imposed on data-analysis will be the reliable identification of the large number of unknown compounds. Biochemical approaches to data interpretation can yield directly testable hypothesis (M. Milburn, 2006), and may be preferable to the pure application of data-mining strategies.

Metabolomics is well suited to a Cloud based workflow, as it is a data and processing intensive scientific undertaking, owing to its essentially large-scale biochemical caliber that explores hierarchies from the bottom-up (R G. Shulman, 2005)[31]. P. Anderson et al from Microsoft Research, recently applied Cloud computing to mine Nuclear Magnetic Resonance spectra in the context of a Metabolomics workflow. (Paul Anderson, 2010)

The aggregator platform can utilize the Cloud by hosting individual or a set of services within a Cloud environment such as Amazon's EC2 compute cloud, which is based on virtualized hardware resources, as opposed to a traditional physical webserver (See 3.5.2.3). Service assignment is transparent and can be performed seamlessly via a central service list. Hosting services in the Cloud may prove practical under computationally demanding scenarios, such as spectra deconvolution, spectra mining, metabolite identification and spectra auto-annotation.

As the service does not put demands on the user-application but only a contract on the data-exchange, remote functionality via an RPC-API can also be used from a workflow environment, that merges local with remote processing, such as the JAVA based *Taverna* (D. Hull, 2006).

Concluding, the advantages of Cloud computing are the collocation of data with computation, reusable software packages and economic benefits in hosting web-services. For the future of the aggregator, elastic scaling of computing resources would allow financially viable service offerings, for the direct processing across many metabolomics datasets at once, generated by different laboratories, with the outlook of applying uniform and centralized data processing routines. The status quo, with differently applied algorithms and presets by different laboratories, may contribute to impeded data interpretation. However such an effort would require that entire raw Metabolomics datasets are made available along with the publication.

5.4 Outlook and future work

[31] http://books.google.com/books?id=ydghP438loYC&pg=PA176 Metabolomics by in vivo NMR

Future revisions of the aggregator portal may see better integration of *Plantmetabolomics.org* (PM) data, particularly in regard to data normalization and processing routines. At the moment PM data is normalized according to each specific laboratory protocol (Bais, 2010).

Interactive pathway maps may be created via *d3.js* and the underlying maps generated from KGML (KEGG Markup language) files which are transformed via an XSLT processor and purpose-written XSL stylesheet, as part of work that is currently in its conceptual stages. The KEGG maps could be merged with maps from *Biocyc* and difference or commonalities highlighted. Pathway comparison between organisms could be rendered interactively in a similar manner to Plantcyc's implementation, with the addition of overlaying metabolite quantities from a given experiment, uploaded by the user. Presenting metabolic networks and pre-computed network parameters in addition to interactive network visualization will remain a significant aim of the Aggregator portal, to serve users with systems biological aspects.

The author would like to state that some of the described accomplishments lie outside the scope of this thesis, and that best care was taken towards citing, but errors do happen. All pictures herein are produced by the author and licensed under CC-BY-SA3 with one exception, present in Chapter 5.3.4 and overtly the very last picture (Fig 36).

This work concludes with a seemingly timeless quote and picture, in reminiscence of the ultimate aim of aggregator portals, as the unbiased assistance of science itself.

…Find the picture that sets out all the known facts in the best arrangement and that therefore has the highest degree of probability. Further, we have to be prepared always for the possibility that each new discovery, no matter what science furnishes it, may modify the conclusions we draw.

Alfred L. Wegener The Origins of Continents and Oceans

Fig 36 The Blue Marble, NASA Images

6. Appendix

6.1 MVC Example

A simple event-based concept with jQuery bind[32] and trigger is presented. The intend of bind is the attachment of user defined event handlers (functions) to elements of the user interface. *Trigger* emits custom events, which are never triggered by the Browser itself.
Note: jQuery is a cross-browser JavaScript library that simplifies Browser functionality.

User interface element:
A button in HTML

```
<button id=”btnben”>Benjamin Button </button>
```

```
function UpdateView() {
    var sql = $('#dataSQL').html();
    ...
   //various logic to update the view according to the event-nature

   $('#viewlog').append('<br>Render call, parameters: ' +
JSON.stringify(arguments));
}

$('#btnben).click(function() {
    $(document).trigger('INPUT_CHANGE');
});

//Defining multiple events
$(document).bind('INPUT_CHANGE SQL_EVENT TABLE_UPDATE', function() {
    UpdateView();
});
```

6.2 DOM Example:

Rewriting the current document

[32] http://api.jquery.com/bind/

```
document.open();
document.write('<html><head></head><body> text content</body></html>');
document.close();
```

Creating a new document

```
var doc = document.implementation.createHTMLDocument("html-document title");
doc.documentElement.innerHTML = 'text content';
//result: see Fig 12
```

A generic document can be generated as follows

```
// document.implementation.createDocument [33] (namespaceURI, qualifiedNameStr,
DocumentType);
var doc = document.implementation.createDocument(null, "rootTag", null)
//inserts tag into a specified names pace e.g. NS =
'http://www.w3.org/1999/xhtml'
var body = document.createElementNS(null, 'body');
doc.documentElement.appendChild(body);
```

```
var html = '<parentElement><childElement>some text content
</childElement></parentElement>'
//a new document doc is created as in the html-document example...
var doc = document.implementation.createHTMLDocument("html-document title");
doc.documentElement.innerHTML = html;

doc.childNodes.item(1).childNodes(1).childNodes[0].childNodes[0].innerHTML
> "some text content "
doc.childNodes[1].childNodes[1].firstChild.firstChild.nodeName
>"CHILDELEMENT"
doc.childNodes[1].childNodes[1].firstChild.firstChild.nodeType
>1
doc.childNodes[1].childNodes[1].firstChild.firstChild.innerText
> "some text content "
```

Notably, in JavaScript the prefix *get* and *set* is absent in the method-name. For instance, in standard DOM APIs the method `firstChild()` would be named – in camelCasing - `getFirstChild()`. Likewise the function call `childNodes(1)` would thus become `getChildNodes(1)`

DOM Nodes can be quickly traversed by heeding three node relations: *parents*, *children* and *siblings* with two

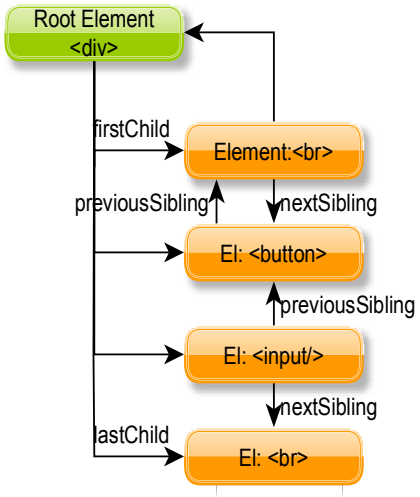


Fig 6 Dom Traversal using three key-relationships

[33] https://developer.mozilla.org/En/DOM/DOMImplementation.createDocum

hierarchical orders (Fig 36), yielding 6 (3x2) functions as listed below.

```
parentNode (parentElement)
childNodes (childElement)
firstChild (firstElementChild)
lastChild ( lastElementChild)
nextSibling (nextElementSibling)
previousSibling (previousElementSibling)
```

Bracketed is the corresponding element's attribute which was formerly used in the non-standard MS Internet Explorer (IE) notation[34]. These should not be referenced anymore. The attribute provided in brackets are MS IE method-aliases. Likewise, instead of using `childElementCount` the use of `childNodes.length` is advised.

DOM-traversal is illustrated in the following example, with a *division-element* comprising a *button* and an *input field*. The selected Element is the button

```
<div>
      <br><button>click me</button> <input/><br>
</div>
```

6.1 ETL Example for extracting web-resources via the DOM:

A useful method developed for *scraping* web-resources is based on *curl* via predefined criteria and subsequent document parsing and extraction using the DOM and XPath - in case of structured web-documents. XPath, provides quick rule based access to XML structured documents. Scripts are available on github[35], and in the Appendix. License notes for the content of a particular *web-resource* must be heeded, but often times *fair use*[36] policies are applicable to academic users.

A fast way of extracting multiple web-resources such as *web-documents* is through utilizing the Browser's DOM, by splitting and replacing the `<body>` tag through `<div id="container#1...n">` and creating a single concatenated HTLM or XML document.

---

[34] http://msdn.microsoft.com/en-us/library/ms534327(VS.85).aspx

[35] https://github.com/lsauer/scripts_n_snips/blob/6df76faa40abdce64525fcb82a049d1521dafb3f/python/ripcage_multithreaded_resripper.py

[36] http://en.wikipedia.org/wiki/Fair_use ; printed also in the appendix

This document is subsequently loaded in a web-browser (for files less than <10MB) or standalone DOM-parser (for files exceeding >10MB), to build up the DOM-tree, which is then easily amenable to trail-and-error test-extractions from within the Browser interface through a WYSIWYG (What You See Is What You Get) –like environment.
The aim of developing testing-routines is typically to store the extracted content in a (fast) serialization-data container such as JSON, XML, Pickle or MessagePack.

Example: split-replacing Markup-Code based on Regular Expressions (exemplary JavaScript code)

```
var cntcontainer = 0; //must be increased with each new processed document
var txtbody = "<html><head><title>...</title></head><body
lang='englishORalien'>....-asdf123- is my favorite pass-
word.</body></html>".split(/<body.+?>|<\/body>/)[1]
//embed the string e.g. txtbody.big() or '<tag id=...>'+ txtbody + '</tag>'
txtbody += '<div id="container"'+cntcontainer+'>'+ txtbody + '</div>';
```

If *pattern-capturing -brackets (...)* are set, the Regular Expression match will be stored and included in the resulting array.

6.2 Using the Project Toolkit and Web-services for web-application development

JSON is a serialization format, based on the JavaScript Object Notation. Remote Procedure Calls are function calls which are invoked at a remote endpoint. In-between, an appropriate handshake and protocol has to be chosen. The simplicity and lack of overhead of the JSON-format, in contrast to XML-based envelopes, make a JSON-RPC ideal for many generic webservices. Envelopes are (meta)-data containers, wrapping the data to be sent or received. These envelopes can be either human readable (JSON, XML), or binary and highly encoded.
A JSON-RPC generally sends an HTTP request to a server hosting webservices with a JSON-RPC compatible handler endpoint.

Following is a code example for a simple web-application, whose purpose is to allow the entering of a *compound name* in order to retrieve *mass*, *chemical formula* and *database crossreference* information.
The resulting application is subsequently investigated from an MVC standpoint.

In this example a complete HTML-document, and all user-interface elements are created programmatically in JavaScript. It is assumed that the *rpcClient.js* and a corresponding css-stylesheet are loaded.

```
var doc = document.querySelector('body');
var ediv,ebtn,eput,divhtml,edivout;

doc.createElement;
divhtml = '<b>Retrieve Crossreference-information for a compound e.g. Acety-
lene </b><br>';
doc.appendChild(ediv = document.createElement('div'))
ediv.innerHTML = divhtml;
ediv.appendChild(ebtn = document.createElement('button'))
ediv.appendChild(eput = document.createElement('input'))
ediv.appendChild(edivout = document.createElement('div'))

//setup elements
edivout.innerText = "result..."
eput.type = "text";
eput.placeholder = "e.g. acetyl";
ebtn.innerHTML = 'click to retrieve a compound';
ebtn.onclick = function(){
      rpc.get_xrefByName(eput.value) >> edivout;
};
```

Alternatively, declarative syntax is used in the following example to put emphasis on the small-size and usage of HTML Editors (Fig 3.2.3). Summarizing, in just three lines of code a small, albeit functional web-application utilizing the Aggregator's web-services has been developed, which allows looking up compounds by their trivial name.

```
<script src="res/js/rpcClient.js" iRPCautoLoad="default"></script>
<div><b>Retrieve XRef Info for a given Compound... </b><br><input/>
<button onclick="rpc.get_xrefByName(eput.value) >> this.nextSibling;">click
to retrieve a compound</button><div></div></div>
```

Note: The iRPCautoLoad attribute, can assume values of *false*, *true*, *default* or a comma separated list of *service-domains* to be included during loading. The value *false* corresponds to omission of the attribute. The browser recreates a valid HTML document by placing the markup-content inside the body-element of a newly instanced *document*, if no HTML-tags are present. Whilst not recommended, the example serves an illustrative purpose.

The code above can be expanded to:

```
<script src="res/js/rpcClient.js"></script>
<script>
var jsonSmd =
"rpc_handler.php?loadwsdl=kegg,eutils,chemspider,massbank,CSMassSpecAPI,csInC
```

```
hI,metlin,bind,pug_soap,pathexplorer&servicecached=kegg,pathexplorer,massbank
,soapCall&comment=0";
//create a new iRPC instance
rpc = new JsonRPC;
//intialize: retrieve and map remote procedures
rpc.Service(jsonSmd);
</script>
<div><b>Retrieve Crossreference-information for a compound e.g. Acetylene
</b><br></div>
<input/><button>click to retrieve a compound</button>
```

The applied stylesheet is provided in the *Sass*-format in the Appendix.

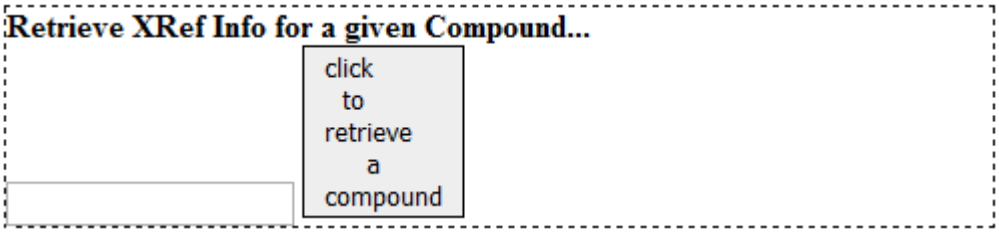


Fig 3.2.3 –Showing the exemplary web-application, from within a WYSIWYG HTL Editor.

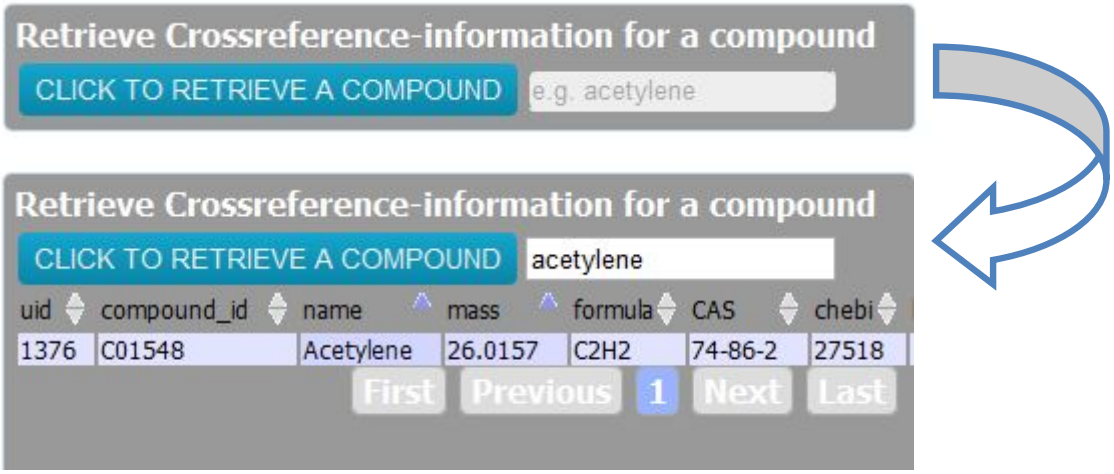


Fig 3.2.4 A Web-application with three lines of code and an applied cascading stylesheet (CSS); With one additional line for the inclusion of *JQueryTable*, the table results are rendered interactively (shown). (Note, the browser recreates a valid HTML document by placing the markup-content inside the body-element)
In this instance, *Datatables*[37] is used for advanced interactive table-functionality

[37] http://datatables.net/ ; Note: A thead row is crucial to be defined: http://www.datatables.net/forums/discussion/6924/using-datatables-without-thead-tag.-perform-sorting-and-filtering-via-dropdown-menu-with-ui-li-tags./p1

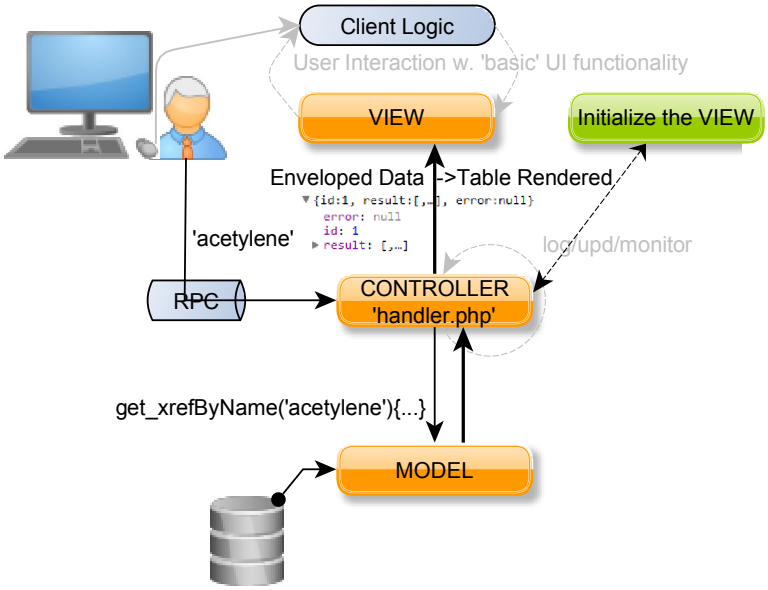
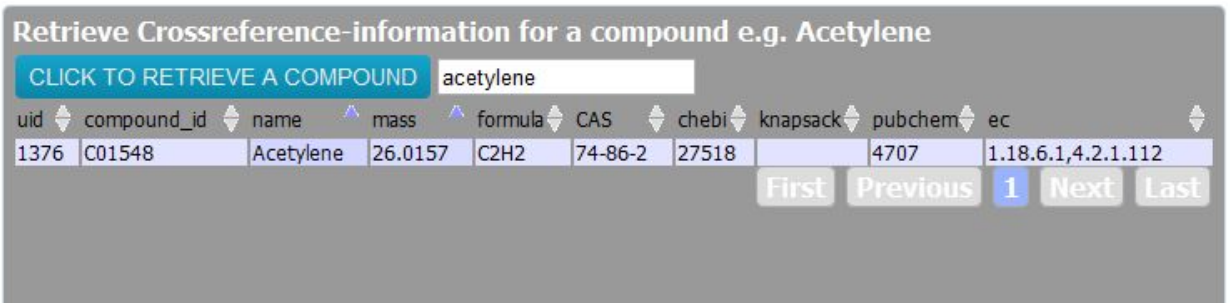


Fig 3.2.5 Case study of the MVC Pattern for exemplary, simple web-application

The following example of 146 characters would fit inside a twitter message. Results are shown on the fly as the user types the chemical compound's name.

```
<script src="rpc.js" iRPCautoLoad="1"></script>
<div>Get XRef:<br><input onchange="rpc.xrefByName(this.value)>>this.Sibling;"
/><div></div></div>
```

Complete Result of the Web-application Example



Following is the Applied StyleSheet, provided in Sass[38], which is more concise than CSS. An interconverter for easy project migration is available[39].

```
body > div
```

[38] http://sass-lang.com/

[39] http://css2sass.heroku.com/ Jose Barrantes

```
  background: #999
  font: bold 1em tahoma
  margin: 10px
  padding: 5px
  color: white
  border: solid 1px rgb(200, 221, 238)
  border-radius: 5px

input.submit:hover, button:hover, .shadowoutline:hover
  -webkit-box-shadow: 0px 0px 20px #555
  -moz-box-shadow: 0px 0px 20px #aaa
  box-shadow: 0px 0px 20px #555
  cursor: pointer

input.submit, button
  color: #EEE
  text-transform: uppercase
  margin-top: 5px
  padding: 5px 8px
  background: -moz-linear-gradient(25% 75% 90deg, #0a85a8, #18a5cc)
  border: none
  font-size: 14px
  -webkit-transition: -webkit-box-shadow 0.3s linear
  -moz-border-radius: 4px
  -webkit-border-radius: 4px
  border-radius: 4px
  background: -webkit-gradient(linear, 0% 0%, 0% 100%, from(#18a5cc),
to(#0a85a8))
  background: -moz-linear-gradient(25% 75% 90deg, #0a85a8, #18a5cc)
```

6.3 Using the Aggregator Webservices

Underpinning the Aggregator frontend, is a software package which is addressable via a RPC interface. This interface is directly accessible within the Browser's interactive JavaScript console. All RPC functionality is provided through the JavaScript object *rpc* per default. For instance KEGG's SOAP web-services are mapped to the *rpc* object on the client. These SOAP services are loaded from a WSDL-function's map by the Aggregator and can be either passed through or extended to provide additional data-consolidation. The developer does not have to heed adherence to a particular interface-contract. Rather, interfaces are generated and managed automatically. To create

and expose a new server-side function all that has to be ensured is that the function is set as *public* according to Object Oriented Programming rules.

To load for instance new remote service functionality which is provided as a SOAP-service, the following commands can be executed:

```
rpc.putWsdlOnServer("http://biomoby.org/services/wsdl/ipb-halle.de/MetFrag_Query",
'metfrag')
rpc._loadwsdl.push("metfrag")
rpc.Remap()
XHR finished loading: "http://localhost:8888/gator/rpc_handler.php".rpcClient.js:1032
loader intervals cleared
...
rpc.metfrag_query_MetFrag_Query()
rpc.metfrag_MetFrag_Query
This function is lazy-loaded! To resolve lazy-references please invoke it like this:
rpc.metfrag_MetFrag_Query("metfrag_X")
Please stand-by whilst the lazy-function references are resolved for
|metfrag_MetFrag_Query| ... Afterwards use _h or Help() to learn how to use the func-
tion. e.g. rpc.metfrag_MetFrag_Query._h
```

To bring up the Browsers JavaScript console press CTRL+SHIFT+J (Google *Chrome*, *Opera*,MS *IE*, *FireFox*). Type *rpc._h* for instance, to get a summary of the running RPC-instance:

```
rpc.Help()
iRPC v0.3
Version of the Service Mapping Description(SMD):2.0
Envelope of the Service Mapping Description(SMD):JSON-RPC-1.0
Transport-Mode:POST
Service-target-url:../gator/rpc_handler.php
Parameters that can be passed to the SMD Service in the form of
"url?paramKey=paramValue&..."; Parameters:#10
        Param#1:    q {string}              default:-;    optional:true
        Param#2:    loadwsdl {string}              default:kegg; optional:true
        Param#3:    servicecached {string}         default:kegg; optional:true
        Param#4:    cacheIgnorelist {string}              default:-;    optional:true
        Param#5:    ignoreErrors {string}          default:-;    optional:true
        Param#6:    comment {string}               default:true; optional:true
        Param#7:    smdextended {string}           default:true; optional:true
        Param#8:    loadlazy {string}              default:true; optional:true
        Param#9:    getsmd {string}         default:kegg; optional:true
        Param#10:    content {string}              default:json; optional:true
 rpcClient.js:1069
undefined
```

Examples

Following are several examples of using provided web-services.

To retrieve pathways through KEGG Orthology-ID's tType
rpc.kegg_get_pathways_by_kos(['ko:K00016', 'ko:K00382'])

To Retrieve the subcellular location of through Arabidopsis gene identifiers type
`rpc.getLocationSUBAByIdAcs('AT3G52880.1')`

Fig 3.2.1 Showing the returned JSON data-set with the sub-cellular localization information of the gene-product from AT3G52880.1

Type `rpc.kegg_bconv("hsa:1798 mmu:13478 dme:CG5287-PA cel:Y60A3A.14")`

```
XHR finished loading: "http://localhost:8888/gator/rpc_handler.php".
rpcClient.js:1010
Result id#38:
hsa:1798      emb:AA833740  equivalent
hsa:1798      emb:AK095122  equivalent
hsa:1798      emb:AK095718  equivalent
hsa:1798      emb:AK128572  equivalent
hsa:1798      emb:AK225994  equivalent
hsa:1798      emb:AK301973  equivalent
hsa:1798      emb:AK312783  equivalent
hsa:1798      emb:BC000325  equivalent
hsa:1798      emb:BC047771  equivalent
hsa:1798      emb:BM982950  equivalent
hsa:1798      emb:BT006802  equivalent
hsa:1798      emb:DA023424  equivalent
hsa:1798      emb:HQ447193  equivalent
hsa:1798      emb:Z82022    equivalent
```

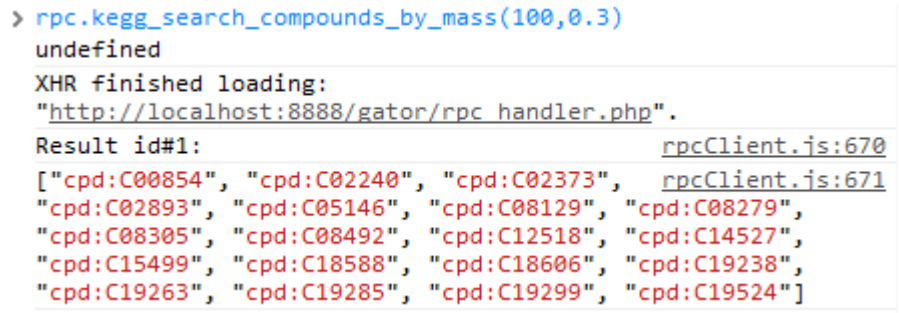


Fig 3.2.2 Showing the returned result of KEGG-DB compounds matching the mass range from 99.7u to 100.3u

When typing for instance *rpc.kegg_* , the autocompletion-feature can be invoked by pressing the TAB key. Picking from the resulting list for instance `rpc.kegg_bconv` and pressing ENTER, provides a basic function description, as follows:

```
Help for the remote method "kegg_bconv":
#List of parameters:
       @param#1: $string {string} optional:false
#Return value-type of the function:
       @return:string
```

The same result can be achieved by entering `rpc.kegg_bconv._h or help()` -the latter method allows passing additional parameters.

A manual, a so-called *man-page* for the function can be brought up. Man-pages are generated on-the-fly and its quality depends on the underlying web-service documentation.

Type `rpc.kegg_bconv.man`

```
...
0002")

btit(string:str)

Wrapper method for btit command. btit is used for retrieving the definitions
by given database entries.  Number of entries given at a time is restricted
up to 100.
Return value:
string
Example:
# Returns the ids and definitions of four GENES entries "hsa:1798",
# "mmu:13478", "dme:CG5287-PA" and cel:Y60A3A.14".
btit("hsa:1798 mmu:13478 dme:CG5287-PA cel:Y60A3A.14")

bconv(string:str)

The bconv command converts external IDs to KEGG IDs.
Currently, following external databases are available.
```

```
External database  Database prefix
-----------------  ---------------
NCBI GI            ncbi-gi:
NCBI GeneID        ncbi-geneid:
GenBank            genbank:
UniGene            unigene:
UniProt            uniprot:
The result is a tab separated pair of the given ID and the converted ID
in each line.
Return value:
string
Example:
# Convert NCBI GI and NCB
...
```

Each further invocation of `rpc.kegg_bconv.man` will 'turn' or rotate to the next page to the next found entry that matches the function-name.
Many webservices, provisioned by major Bioinformatics groups, are mapped to the RPC interface by default. Additionally specific Aggregator services fill functional niches such as the conversion of SMILES to the SVG vector graphics format.

In order to efficiently load tens of thousands of RPC methods (procedures), with virtually instantaneous loading times, service procedures are mapped sparsely. Upon first invocation of a sparsely mapped RPC function, an SMD-file for a *service-domain* is requested and loaded which leads to the Service-provider remapping the sparsely mapped-domain and subsequently passing through of the RPC-call. This incurs a certain delay at the first invocation of an *unmapped* or sparsely mapped function. An additional time-delay is not incurred when the *performance-mode flag* is set. Yet this requires the service-domain to be passed as a function argument and provides no parameter checking or help. This mode can be set as follows:
rpc.addMode = rpc.MODE.PERFORM;

Typing `rpc.getMode` provides an overview of the RPC-state.

```
Mode {UInt32} is:552 [1000101000]
        [0000000000000000] #OFF     -> false
        [0000000000000001] #SYNC    -> false
        [0000000000000010] #OFFLINE -> false
        [0000000000000100] #RESERVED-> false
        [0000000000001000] #CHAIN   -> true
        [0000000000010000] #RETBYREF-> false
        [0000000000100000] #PARSE   -> true
        [0000000001000000] #GLOBAL  -> false
        [0000000010000000] #PERFORM -> false
        [0000000100000000] #LOADING -> false
        [0000001000000000] #LOADED  -> true
 rpcClient.js:442
552
```

More information is provided in the documentation of the iRPC project description on github.

The following example shows a simple call to a kegg_service via SOAP with the intend of finding masses of 100u +/- 0.3u. The function can be extended on the server-side. Note that SOAP is used because of the way KEGG web services are implemented. RESTful, RDF, XML-RPC, JSON-RPC services can be used as well, and are abstracted to the uniform JSON-RPC Interface , relying on JSON-XHR or websockets.

6.4 Service Map/Mapping Description -SMD

SMD [40]is a lightweight alternative to XML-SOAP based WSDL (Web Service Definition Language), which for many applications suffices at the benefit of increases speed.

"A Service Mapping Description (SMD) is a JSON representation describing web services. An SMD can define the various aspects of a web service such that clients can coherently interact with the web services. An SMD can be used by generic tools to generate interfaces, human and programmatic, to access available web services. A wide array of web services can be described with SMD including REST services and JSON-RPC services. The SMD format is designed to be flexible, compact, simple, readable, and easily implemented."[41]

The original SMD data-fields were extended by the field '*warning'* and the field *error*. The field *description* contains comments generated from the server-script's comments in HereDoc notation. This is facilitated via PHP's reflection capabilities. Any PHP *public function* in the class *rpcServerFunctions* which is added in the *jsonRPCServer.php* is automatically exposed via the RPC *handler*-script. The client-RPC implements functionality that facilitates the forced reloading of parts of function-maps and for remapping of these functions. Server-side invocation of the client's SMD remapping, could be performed through a PUSH event based on the server-sent events(SSE)[42] API of current browsers, when a difference in the SMD's length is detected (i.e. a difference or *diff*). SSE is part of an ongoing HTML 5 draft.
This concept allows concurrent revision of functions, which may prove particular, useful during live-debugging in a productive-environment. For instance, upon the firing of a dedicated server-sent event all 'connected' client's would remap parts of their RPC-

[40] http://www.google.com/url?q=http%3A%2F%2Fgroups.google.com%2Fgroup%2Fjson-schema%2Fweb%2Fservice-mapping-description-proposal
[41] http://livedocs.dojotoolkit.org/dojox/rpc/smd
[42] http://dev.w3.org/html5/eventsource/

functions. Notably, functions for which only the *function-procedure* in the server-code changes, can be revised seamlessly during bug-fixing, without any impact on the client-side.

Example SMD-file loaded upon a GET request to handler.php

JSON formatting and validation was performed via Google's Chrome V8 JavaScript engine and Jsonformatter[43]

The RPC- envelope specifies the data-container and version. JSON is set as default, XML can be set alternatively. Messagepack will be supported in a future *iRPC* revision. Tab X lists parameters, parameter-data type and a description of parameters, which can be sent via an GET-HTTP request to the RPC-handler (handler.php) to correspondingly result in different SMD outputs.

The parameters are as follows:

Tab X – List of Parameters of the RPC service-endpoint (handler.php)

Parameter	Description	Type
isDebug	provides additional debug messages	Bool
q	: search for specific functions only and retrieve their mappings	String
wsdls	names of local wsdl files whose functions can be put in the SMD	String1,str2
loadwsdl	sets the names of SOAP services which are to be cached, delimited by comma ',' ;	String1;str2
servicecached	TODO: sets a list of operations to be cached, accepts regex	String1;str2
cacheIgnorelist		
ignoreErrors		Bool
comment	turns off comments in the SMD-JSON, for parser compatibility w. e.g. jquery	Bool
smdextended	Whether the SMD 2 specification should be extended or adhere to *strict*	Bool
loadlazy	whether SMD service-information of services not on the default-load list should be loaded lazily; bool	Bool

[43] http://jsonformatter.curiousconcept.com/#jsonformatter

getsmd	list of wsdl,... etc services which should be returned in full //differs from loadwsdl in that loadlazy is ignored, and no mapping of PHP functions are output; default RPC_DEFAULT_LOAD_WSDL	string
content	MIME-contenttype that should be set; useful also during debugging -> choose text	Enum: [json|text|js]

In example X, which is abbreviated, a warning is set intentionally to demonstrate the error reporting JSON-payload. Warnings and errors are highlighted in the Client's Java-Script console, to aid developers during debugging.

Example X


```
{
   "warning":"{\"datetime\":\"2011-12-
21 20:43:50 (CET)\",\"errornum\":8,\"errortype\":\"Notice\",\"errormsg\":\"Try
ing to get property of non-
object\",\"scriptname\":\"jsonRPCServer.php\",\"scriptlinenum\":886}\n\n\r",
   "SMDVersion":"2.0",
   "additionalParameters":true,
   "envelope":"JSON-RPC-1.0",
   "parameters":[
      {
         "default":"kegg",
         "name":"loadwsdl",
         "optional":true
      },
      {
         "default":"kegg",
         "name":"servicecached",
         "optional":true
      },
      {
         "default":"",
         "name":"cacheIgnorelist",
         "optional":true
      },
      {
         "default":false,
         "name":"ignoreErrors",
         "optional":true
      },
   ],
      "bind_BIVGetComplexRecord":null,
      "bind_BIVGetInteractionRecord":null,
      "bind_BIVGetNeighbours":null,
      "bind_BIVGetRecord":null,
      "bind_isServiceAlive":null,
      "bind_textSearch":null,
      "bind_textSearchAttachment":null,
      "chemspider_AsyncSimpleSearch":null,
      "chemspider_GetRecordDetails":null,
      "chemspider_GetRecordImage":null,
      "chemspider_SimpleSearch":null,
      "chemspider_SimpleSearch2IdList":null,
```

```
      "chemspider_StructureSearch":null,
      "chemspider_SubstructureSearch":null,
      "getXpathResult":{
         "description":"",
         "parameters":[
            {
               "name":"url",
               "optional":true,
               "type":"string"
            },
            {
               "name":"path",
               "optional":true,
               "type":"string"
            }
         ]
      },
      "metlin_FragmentSearch":null,
      "metlin_MetaboliteSearch":null,
      "metlin_NameSearch":null,
      "multiply":{
         "transport": "POST",
         "envelope": "JSON-RPC-2.0",
         "target": "http://127.0.0.1:8118/handler/ ",
         "description":"Multiplies a series of numbers.",
         "parameters":[
            {
               "name":"num1",
               "optional":true,
               "type":"integer"
            },
            {
               "name":"num2",
               "optional":true,
               "type":"integer"
            },
            {
               "name":"etc",
               "optional":true,
               "type":"integer"
            }
         ]
      }

   },
   "target":"rpc_handler.php",
   "transport":"POST"
}
```

Note: In example X, some services are undefined (‘empty’), with the value set to `null` when the RPC-option *smdextended* is set (Tab X), otherwise the value is set to an empty object `{};` In the SMD Proposal Specification, all parameters are optional; The empty object i.e`. {}` is assigned the meaning of undefined parameters. However, an SMD function-name which is mapped to `null`, is defined as a sparse-function and reserved for lazy-loading. In this instance, the function is not mapped until an SMD-subset for a given service-domain is requested from the service-endpoint, containing definitions for formerly undefined RPC-functions (for a given service-domain). A domain encompasses

a set of services whose domain-name consists of an alphanumeric string between 2 to 25 characters and is prefixed by an underscore to the function-name. For instance the rpc-function `chemspider_SimleSearch` has *chemspider* as the domain. Programmatically speaking, the service domain is the substring from position 0 to the first Index of any underscore.
The function *multiply* overrides the envelope, service-endpoint and parameter transport method, in compliance with the SMD-proposal specification. The *variable declaration of* further *endpoints* is part of the author's *SMD-Extensions*, introduced in this project. Defining a different endpoint allows client-side service-discovery and re-/mapping or rerouting of workflows. A workflow is a defined chronological list of RPC calls – which can be nested -, with data being passed between RPC-functions within the list.

Lazy loading features a *granularity-tier*, which can be set by the performance-mode of the client RPC instance. In performance mode all unmapped functions point to the same helper function, to which the domain-name must be passed in order for loading and remapping of the lazy-loaded functions. (JavaScript, unlike PHP, lacks powerful reflection methods, and cannot retrieve its own variable names). Performance mode is not generally recommended. Yet in principle, tens of thousands of functions can be mapped with ease and low impact on overall loading time, with general foresight of future distributed computing experiments.
When the HTTP-Content payload is *gzip*-compressed, an SMD with 100.000 functions, and an average name-length of 20 characters, yields an SMD file-size of typically 200-300kB and takes less than a second to map on a modern computer (2GHz Intel Core Duo).

For SMD's loaded from a different domain than the server of the web-application, cross-site-scripting (XSS) policies apply, which would restrict any significant interaction between local functions and functions loaded from another domain. Hence *JSONP* is used in this scenario, with a static-callback function named after the name of the SMD, or by passing the url-parameter *jpfn*.

6.5 Database related

Mysql SELECT Syntax

```
SELECT
    [ALL | DISTINCT | DISTINCTROW ]
      [HIGH_PRIORITY]
      [STRAIGHT_JOIN]
```

```
      [SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
      [SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]
    select_expr [, select_expr ...]
    [FROM table_references
    [WHERE where_condition]
    [GROUP BY {col_name | expr | position}
      [ASC | DESC], ... [WITH ROLLUP]]
    [HAVING where_condition]
    [ORDER BY {col_name | expr | position}
      [ASC | DESC], ...]
    [LIMIT {[offset,] row_count | row_count OFFSET offset}]
    [PROCEDURE procedure_name(argument_list)]
    [INTO OUTFILE 'file_name'
        [CHARACTER SET charset_name]
        export_options
      | INTO DUMPFILE 'file_name'
      | INTO var_name [, var_name]]
    [FOR UPDATE | LOCK IN SHARE MODE]]
```

Selected Database Tables

```
DROP TABLE IF EXISTS `kegg`.`icmpds`;
CREATE TABLE  `kegg`.`icmpds` (
  `uid` int(11) NOT NULL,
  `compound` varchar(255) NOT NULL DEFAULT 'Unknown',
  `inchi` mediumtext NOT NULL,
  `moldata` mediumtext,
  `specdata` blob,
  `casno` varchar(20) NOT NULL,
  `barcode` int(20) NOT NULL,
  `derivative` varchar(20) NOT NULL,
  `RT` varchar(20) NOT NULL,
  `scan_range` varchar(25) NOT NULL DEFAULT '0,',
  `mz` double NOT NULL,
  `charge` int(11) NOT NULL,
  `score` float NOT NULL,
  `adduct` varchar(20) NOT NULL,
  `metadata` tinytext NOT NULL,
  `experiment` tinytext NOT NULL,
  `relations` tinytext NOT NULL,
  `serialized` tinytext NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COMMENT='internal chemical database';
```

Proposed table for storing single compound spectra

The fields 'serialized', 'metadata' and 'experiment' are meant to hold serialized data.

6.6 Selected Scripts, Files and other Resources

6.6.1 Chemical Compound API Interface Definition for Third party providers in IDL

```
struct XRef {
    string provider;
    string entryid;
};

interface MetLCEntity {
      //Internal Id
      int32 MetId;
      //IUPAC name
      string MetName;
      //quantifier of the entitiy e.g. intensity
      int32 Quant;
      //relational quantifier of the entitiy e.g. intensity in relation to
e.g. log2 ratio
      int32 QuantRel;
      //numeric reference to the experiment-entry, not yet outlined
      int32 ExpId;
      int32 RetTime;
      int32 RetIdx;
      //hash list containing crossreferences for this entry e.g. {KEGG:
C00123, PUBCHEM: ....}
      sequence<XRef> xref;
}

interface MetList {
      //contains the supposed number of entries that were transmitted
      const short Length;
      //dataset identifier e.g. SALK_074697 E2-1
      string Dataset;
      //List entries
      sequence<MetLCEntity> Xref;
      //serialized, experimental-metadata
      string ExpMetadata;
      //#missing values
      int32 MisVal;

      //optional function; retrieves an IDL definition, for dynamic adaption
of interfaces
      getDefinition(in string method, in string version);
};
```

6.6.2 Spectrum used in the Aggregator Spectrum Search Example

```
NAME: Uridine 5'-diphospho-N-acetylglucosamine; GC-EI-TOF; MS; n TMS; BP_73
Apex Scan: 10110
Hit Probability: 7812
Hit Similarity: 958,000000
Mass: 73
Height: 52456
Area: 0,000000
Signal2Noise: 353,725298
Noise: 296,591736
Purity: 0,566882
Profile Purity: 0
Deconvolution Purity: 0
NAME: Arabinonic acid-1,4-lactone (3TMS)_M001186_A165003-101-
x1[NA]_NA_1636,03_TRUE_VAR5_ALK
Num Peaks: 561
  40   0;  41  13;  42   4;  43  39;  44  25;  45 207;  46  10;  47  36;  48   0;  49   0;
  50   0;  51   3;  52   2;  53   3;  54   1;  55  19;  56   4;  57   8;  58  12;  59  75;
  60   7;  61  17;  62   0;  63   0;  64   0;  65   0;  66   7;  67   2;  68   2;  69  36;
  70   3;  71   5;  72  15;  73 998;  74  79;  75 164;  76  10;  77   7;  78   0;  79   0;
  80   0;  81   0;  82   0;  83   2;  84   0;  85   5;  86   0;  87   8;  88   1;  89  36;
  90   2;  91   0;  92   0;  93   0;  94   1;  95   0;  96   0;  97   1;  98   0;  99   5;
 100   0; 101  22; 102  57; 103  55; 104   6; 105   2; 106   0; 107   0; 108   0; 109   0;
 110   0; 111   3; 112   0; 113   5; 114   0; 115   8; 116   6; 117 148; 118  13; 119  12;
 120   0; 121   0; 122   0; 123   0; 124   0; 125   0; 126   0; 127   2; 128   1; 129  23;
 130  43; 131  21; 132   6; 133  62; 134   7; 135   7; 136   0; 137   0; 138   0; 139   0;
 140   0; 141   0; 142   0; 143   8; 144   1; 145   5; 146   0; 147 116; 148  16; 149  29;
 150   4; 151   1; 152   0; 153   0; 154   0; 155   0; 156   0; 157   2; 158   0; 159   2;
 160   0; 161   0; 162   0; 163   0; 164   0; 165   0; 166   0; 167   0; 168   0; 169   1;
 170   0; 171   0; 172   0; 173   1; 174   0; 175   1; 176   0; 177   1; 178   0; 179   0;
 180   0; 181   0; 182   0; 183   0; 184   0; 185   0; 186   0; 187   1; 188   0; 189  38;
 190   6; 191   9; 192   1; 193   0; 194   0; 195   0; 196   0; 197   0; 198   0; 199   0;
 200   0; 201   0; 202   0; 203  14; 204  39; 205  12; 206   4; 207   1; 208   0; 209   0;
 210   0; 211   0; 212   0; 213   0; 214   0; 215  28; 216   4; 217  54; 218  12; 219   6;
 220   1; 221   1; 222   0; 223   0; 224   0; 225   0; 226   0; 227   0; 228   0; 229   0;
 230   1; 231  34; 232   7; 233   3; 234   0; 235   1; 236   0; 237   0; 238   0; 239   0;
 240   0; 241   0; 242   0; 243   0; 244   1; 245   0; 246  25; 247   5; 248   1; 249   0;
 250   0; 251   0; 252   0; 253   0; 254   0; 255   0; 256   0; 257   0; 258   0; 259  17;
 260   3; 261   2; 262   0; 263   0; 264   0; 265   0; 266   0; 267   0; 268   0; 269   0;
 270   0; 271   0; 272   0; 273   0; 274   0; 275   0; 276   0; 277   0; 278   0; 279   0;
 280   0; 281   0; 282   0; 283   0; 284   0; 285   0; 286   0; 287   0; 288   0; 289   0;
 290   0; 291   0; 292   0; 293   0; 294   0; 295   0; 296   0; 297   0; 298   0; 299   0;
 300   0; 301   0; 302   0; 303   0; 304   0; 305   0; 306   0; 307   0; 308   0; 309   0;
 310   0; 311   0; 312   0; 313   0; 314   0; 315   0; 316   0; 317   0; 318   0; 319   0;
 320   0; 321   0; 322   0; 323   0; 324   0; 325   0; 326   0; 327   0; 328   0; 329   0;
 330   0; 331   0; 332   0; 333   0; 334   0; 335   0; 336   0; 337   0; 338   0; 339   0;
 340   0; 341   0; 342   0; 343   0; 344   0; 345   0; 346   0; 347   0; 348   0; 349   3;
 350   0; 351   0; 352   0; 353   0; 354   0; 355   0; 356   0; 357   0; 358   0; 359   0;
 360   0; 361   0; 362   0; 363   0; 364   5; 365   1; 366   0; 367   0; 368   0; 369   0;
 370   0; 371   0; 372   0; 373   0; 374   0; 375   0; 376   0; 377   0; 378   0; 379   0;
 380   0; 381   0; 382   0; 383   0; 384   0; 385   0; 386   0; 387   0; 388   0; 389   0;
 390   0; 391   0; 392   0; 393   0; 394   0; 395   0; 396   0; 397   0; 398   0; 399   0;
 400   0; 401   0; 402   0; 403   0; 404   0; 405   0; 406   0; 407   0; 408   0; 409   0;
 410   0; 411   0; 412   0; 413   0; 414   0; 415   0; 416   0; 417   0; 418   0; 419   0;
 420   0; 421   0; 422   0; 423   0; 424   0; 425   0; 426   0; 427   0; 428   0; 429   0;
 430   0; 431   0; 432   0; 433   0; 434   0; 435   0; 436   0; 437   0; 438   0; 439   0;
 440   0; 441   0; 442   0; 443   0; 444   0; 445   0; 446   0; 447   0; 448   0; 449   0;
 450   0; 451   0; 452   0; 453   0; 454   0; 455   0; 456   0; 457   0; 458   0; 459   0;
 460   0; 461   0; 462   0; 463   0; 464   0; 465   0; 466   0; 467   0; 468   0; 469   0;
 470   0; 471   0; 472   0; 473   0; 474   0; 475   0; 476   0; 477   0; 478   0; 479   0;
 480   0; 481   0; 482   0; 483   0; 484   0; 485   0; 486   0; 487   0; 488   0; 489   0;
```

```
490   0; 491   0; 492   0; 493   0; 494   0; 495   0; 496   0; 497   0; 498   0; 499   0;
500   0; 501   0; 502   0; 503   0; 504   0; 505   0; 506   0; 507   0; 508   0; 509   0;
510   0; 511   0; 512   0; 513   0; 514   0; 515   0; 516   0; 517   0; 518   0; 519   0;
520   0; 521   0; 522   0; 523   0; 524   0; 525   0; 526   0; 527   0; 528   0; 529   0;
530   0; 531   0; 532   0; 533   0; 534   0; 535   0; 536   0; 537   0; 538   0; 539   0;
540   0; 541   0; 542   0; 543   0; 544   0; 545   0; 546   0; 547   0; 548   0; 549   0;
550   0; 551   0; 552   0; 553   0; 554   0; 555   0; 556   0; 557   0; 558   0; 559   0;
560   0; 561   0; 562   0; 563   0; 564   0; 565   0; 566   0; 567   0; 568   0; 569   0;
570   0; 571   0; 572   0; 573   0; 574   0; 575   0; 576   0; 577   0; 578   0; 579   0;
580   0; 581   0; 582   0; 583   0; 584   0; 585   0; 586   0; 587   0; 588   0; 589   0;
590   0; 591   0; 592   0; 593   0; 594   0; 595   0; 596   0; 597   0; 598   0; 599   0;
600   0;
```

6.6.3 KEGG Color Codes (Kanehisa Laboratories, 2012)

Functional category (used in global pathway maps and genome maps)

Carbohydrate metabolism	#0000ee
Energy metabolism	#9933cc
Lipid metabolism	#009999
Nucleotide metabolism	#ff0000
Amino acid metabolism	#ff9933
Metabolism of other amino acids	#ff6600
Glycan biosynthesis and metabolism	#3399ff
Metabolism of cofactors and vitamins	#ff6699
Metabolism of terpenoids and polyketides	#00cc33
Biosynthesis of other secondary metabolites	#cc3366
Xenobiotics biodegradation and metabolism	#ccaa99
Genetic information processing	#ffcccc
Environmental information processing	#ffff00
Cellular processes / Organismal systems	#99cc66

KEGG PATHWAY

Organism-specific pathway	#bfffbf
Reference pathway (KO, EC, Reaction)	#bfbfff
KEGG module	#ffbfbf

Pathway mapping of two organisms (global map)

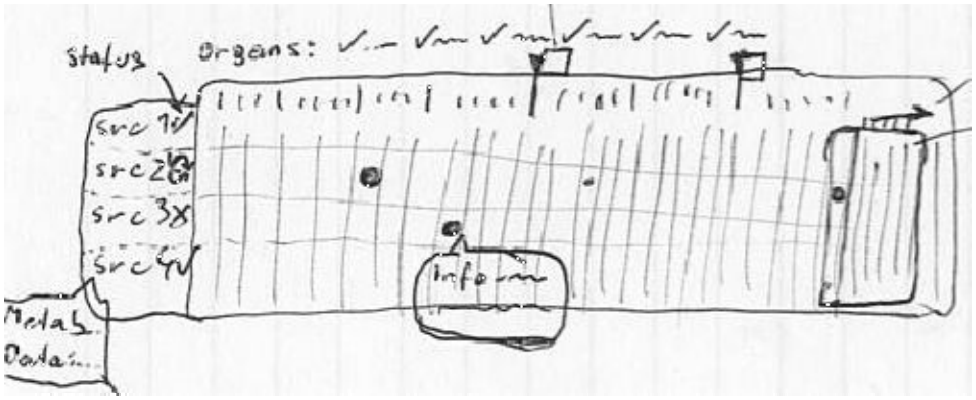
Organism 1	#00cc33
Organism 2	#ff3366

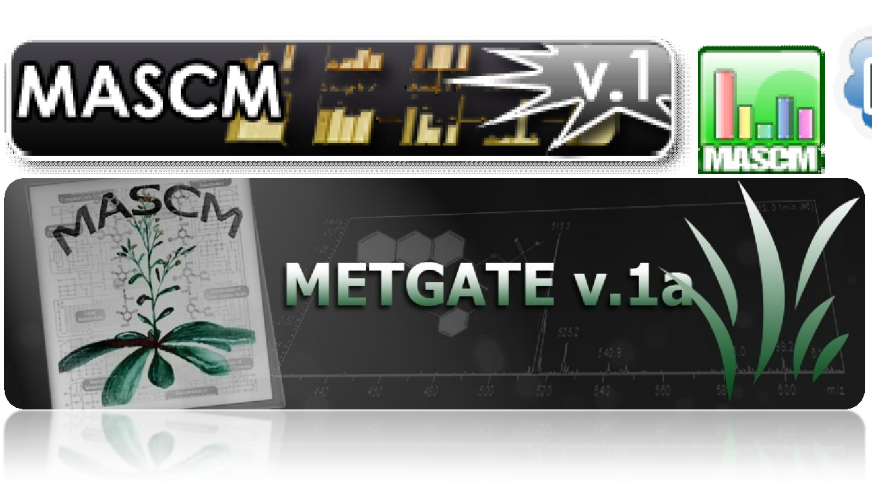
Organisms 1 and 2 #3366ff

Note: the data is available as JSON and XML on the author's gist account.

6.6.4 Designs and miscellaneous resources

Further resources are available on *github.com* . CC-BY-SA 3 applies.





The author's blog features related resources, procedures and tutorials:
http://www.lsauer.com

twitter: @sauerlo
Github: github.com/lsauer
Snippets: gist.github.com/lsauer

7. Bibliography

A Li. (2006). Facing the Challenges of Data Integration in Biosciences. *Engineering Letters, 13:3, EL_13_3_13* .

A Manjunatha, A. R. (2010). Getting Code Near the Data: A Study of Generating Customized Data Intensive Scientific Workflows with Domain Specific Language. *2nd IEEE International Conference on Cloud Computing Technology and Science.* IEEE.

A Smith, M. B. (2011). iology and data-intensive scientific discovery in the beginning of the 21st century. *OMICS Journal , 2011* .

A. Manjunatha, A. R. (2010). Cloud Based Scientific Workflow for NMR Data Analysis. *18th Annual International Conference on Intelligent Systems for Molecular Biology (ISMB).* ISBM.

ACS. (2012). *Chemical Abstracts Service*. Retrieved from http://www.cas.org/index.html

AG McDonald, S. B. (2007). ExplorEnz: a MySQL database of the IUBMB enzyme nomenclature. *BMC Biochemistry 2007, 8:14* .

Akiyama K, C. E. (2008). PRIMe: a Web site that assembles tools for metabolomics and transcriptomics. *In Silico Biol. 2008;8(3-4):339-45.*

Ashburner M, B. C.-T. (2000). Gene ontology: tool for the unification of biology. The Gene Ontology Consortium. *Nat Genet. 2000 May;25(1):25-9.*

Avraham S, T. C. (2008). The Plant Ontology Data-base: a community resource for plant structure and developmental stages controlled vocabulary and annotations. *Nucleic Acids Res 36:D449–D454* .

B Fran, A. J. (2003). *Grid Computing: Making The Global Infrastructure a Reality.* Wiley.

B François, N. M.-A. (2008). Bio2RDF: towards a mashup to build bioinformatics knowledge systems. *J Biomed Inform 41 (5): 706–16* .

B. Youakim. (2008). Service-Oriented Workflow. *Journal of Digital* .

BA Devlin, P. M. (1988). An architecture for a business and information system. *IBM Systems Journal, Volume 27, Number 1, Page 60 (1988)* .

Bais, P. M. (2010). PlantMetabolomics.org: a web portal for plant metabolomics experiments. *Plant Physiology, p. 1807, vol. 152, (2010), pp.109* .

Bolton E, W. Y. (2008). PubChem: Integrated Platform of Small Molecules and Biological Activities. *Annual Reports in Computational Chemistry, Volume 4, ACS* .

Carbon S, I. A. (2009). AmiGO: online access to ontology and annotation data. *Bioinformatics 25: 288–289* .

Chikashi Nobata, P. D. (2011). Mining metabolites: extracting the yeast metabolome from the literature. *Metabolomics, Vol. 7, No. 1. (1 March 2011), pp. 94-101,* .

Christie, W. (1989). *Gas chromatography and lipids.* Oily Press Ltd.

Claudia Brkemeyer, J. K. (2007). Cpt 4 Design of Metabolite Recovery. In *Concepts in plant metabolomics.* Springer.

Critchlow, T. (2001). *Report on XEWA-00: The XML Enabled Wide-Area Searches for Bioinformatics Workshop.* Center for Applied Scientific Computing - Lawrence Livermore National Laboratory (LLNL).

D Reynolds. (2004). *Semantic information portals.* WWW-2004.
D. C. Frost, C. A. (1955). Studies of the Ionization of Molecules by Electron Impact. *Mathematical and Physical Sciences, Vol. 232, No. 1189 (Oct. 25, 1955), pp. 227-235* .
D. Hull, K. W. (2006). Taverna: a tool for building and running workflows of services. *Nucleic Acids Research, vol. 34* .
Data-Archive, R. (2011). *RIKEN Data-Archive*. Retrieved 2011, from http://prime.psc.riken.jp/archives/
David Weisman, H. L.-C. (2011). Novel Computational Identification of Highly Selective Biomarkers of Pollutant Exposure. *Environ. Sci. Technol., 2011, 45 (12), pp 5132–5138* .
DG, R. (2005). Metabonomics in toxicology: a review. *Tox-icol. Sci. 85 (2): 809–22.*
DOI. (2011). *DOI® News, April 2011: 1. DOI System exceeds 50 million assigned identifiers*. Retrieved 2011, from http://www.doi.org/news/DOINewsApr11.html#1
E Schwarz, E. F. (2012). Biomarker discovery in human cerebrospinal fluid: the need for integrative metabolome and proteome databases. *Genome Medicine 2012, 4:39* .
EF Codd. (1970). A Relational Model of Data for Large Shared Data Banks. *Communications of the ACM 13 (6): 377–87* .
Elsevier. (2012). *ClinicalKey and OvidMD: The Importance of End-User Research and Semantic Technologies*. Retrieved from http://elsevier.co.za.dedi166.cpt1.host-h.net/newsworthy/view/clinicalkey-and-ovidmd-the-importance-of-end-user-research-and-semantic-tec
F Azuaje, J. D. (2005). *Data analysis and visualization in genomics and proteomics.* Wiley.
F Crick. (1970). Central dogma of molecular biology. *Nature 227 (5258): 561–3* .
F Pietra. (2002). Evolution of the secondary metabolite versus evolution of the species. *Pure Appl. Chem. 74:2207-2211* .
Fang L., H. Y. (2006). Building web services for scientific grid applications. *IBM Journal of Research and Development Volume:50 Issue:2.3* , 249 - 260.
FM Afendi, e. a. (2011). KNApSAcK Family Databases: Integrated metabolite-plant species databases for multifaceted plant researches. *Plant Cell Physiol (2011), 10.1093/pcp/pcr165* .
Foster, C. K. (1999). Grid computing.
Fox JA, B. S. (2005). The Bioinformatics Links Directory: a compilation of molecular biology web servers. *Nucleic Acids Res. 2005 Jul 1;33(Web Server issue):W3-24* .
Garcia-Hernandez M, B. T. (2002). TAIR: a resource for integrated Arabidopsis data. *Funct Integr Genomics. 2002 Nov;2(6):239-53* .
Gavish, B. (2010). Hard Limits on the Growth of the Internet and Computing Capacity. *International Journal of Information Science and Management* .
Gorsuch, L. L. (1991). *Plants for Toxicity Assessment, Volume 2.* ASTM Committee E-47 on Biological Effects and Environmental Fate.
H Haas. (2003). *Service-Oriented Architecture - W3C Talks*. Retrieved 2012, from http://www.w3.org/2003/Talks/0521-hh-wsa/slide5-0.html
H Theo, R. A. (1983). Principles of Transaction-Oriented Database Recovery. *ACM Computing Surveys 15 (4): 287–317* .
H. Horai, M. A. (2010). MassBank: A public repository for sharing mass spectral data for life sciences. *J. Mass Spectrom., 45, 703-714* .

Haen, R.-d. (2010). *Derivatization of Drug Substances with MSTFA.* Fluka Analytix.
Heazlewood JL, D. P. (2008). PhosPhAt: A Database of phosphorylation sites in Arabidopsis thaliana and a plant specific phosphorylation site predictor. *Nucleic Acids Research 36: D1015-D1021* .
Heller, M. (2007). *REST and CRUD: the Impedance Mismatch.* Retrieved from InfoWorld: http://www.infoworld.com/d/developer-world/rest-and-crud-impedance-mismatch-927
Hernandez T., K. S. (2004). Integration of biological sources: current systems and challenges ahead. *SIGMOD Rec., 33(3), 51-60.*
HJ Joshi, M. H.-H. (2011). MASCP Gator: An Aggregation Portal for the. *Plant Physiology Vol. 155, pp. 259–270,* .
Hummel J, N. M. (2007). ProMEX: a mass spectral reference database for proteins. *BMC Bioninformatics 8: 216* .
InChi, N. (2011). *IUPAC International Chemical Identifier*. Retrieved 2011, from IUPAC Project 2000-025-1-800: http://www.iupac.org/home/publications/e-resources/inchi.html
InChi-Trust. (2011). Retrieved from InChi Trust: http://www.inchi-trust.org/
J G Bundy, e. a. (2008). 'Systems toxicology' approach identifies coordinated metabolic responses to copper in a terrestrial non-model invertebrate, the earthworm Lumbricus rubellus. *BMC Biology 2008, 6:25* .
J Heaton. (2002). Programming Spiders, Bots, and Aggregators in Java. *Sybex* .
J Skolnick, J. F. (2000). From genes to protein structure and function: novel applications of computational approaches in the genomic era. *Trends in biotechnology, 2000 - 144.206.159.178* .
JAOO, Å. (2003). The Model-View-Controller (MVC) Its Past and Present. *JavaZONE, Oslo, 2003.*
Jiang K, H. Y. (2011). A method of biological pathway similarity search using high performance computing. *Conf Proc IEEE Eng Med Biol Soc. 2009;2009:4933-6* .
JISC. (2010). *Virtual Research Environment*. Retrieved from http://www.myexperiment.org/
JL Heazlewood, R. E.-F. (2007). SUBA: the Arabidopsis Subcellular Database. *Nucleic Acids Research, Vol35, Ed 1* , Pp. D213-D218.
Joyent Inc. (2011). *Node.JS*. Retrieved 2011, from http://nodejs.org/about/
JS Morris, P. J. (2007). Analysis of Mass Spectrometry Data Using Bayesian Wavelet-Based Functional Mixed Models. *Biometrics. 2008 June; 64(2): 479–489.*
JY Chung, G. F. (2005). Proceedings of the first interntional workshop on design of service oriented applications. *IBM Reserach Report - WDSOA '05* , 93.
K Degtyarenko, P. d. (2007). ChEBI: a database and ontology for chemical entities of biological interest. *Nucleic Acids Res. 2008 January; 36(Database issue)* .
K Schomburg, H.-C. E. (2010). From Structure Diagrams to Visual Chemical Patterns. *J. Chem. Inf. Model., 2010, 50 (9), pp 1529–1535* .
Kanehisa Laboratories. (2012). *KEGG Color Codes.* Retrieved from KEGG Color Codes: http://www.genome.jp/kegg/kegg1c.html
Kanehisa M, G. S. (2000). KEGG: kyoto encyclopedia of genes and genomes. *Nucleic Acids Res. 2000 Jan 1;28(1):27-30* .

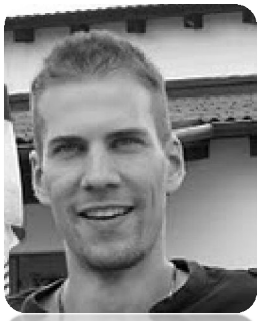
Kanehisa, M. (1997). A database for post-genome analysis. *Trends in Genetics, Vol 13 Nr 9, Sep 1997, pg 375–376* .
KD Schewe, B. T. (2008). emantics in Data and Knowledge Bases: Third International Workshop, SDKB 2008. Springer.
Kell, D. (2007). Metabolomic biomarkers - search, discovery and validation. *Expert Rev. Mol. Diagn. 7(4), 2007* , 2.
Kind T, S. M. (2009). ow Large Is the Metabolome? A Critical Analysis of Data Exchange Practices in Chemistry. *PLoS ONE 4(5): e5440. doi:10.1371/journal.pone.0005440* .
Kresge N, S. R. (2009). 100 years of biochemistry and molecular biology. The decade-long pursuit of a reconstituted yeast transcription system: the work of Roger D. Kornberg. *J Biol Chem. 2009 Oct 23;284(43):e18-20.*
Kwon, H. (2006). Discovery of new small molecules and targets towards angiogenesis via chemical genomics approach. *Curr Drug Targets. 2006 Apr;7(4):397-405.*
L Richardson, S. R. (2007). RESTful Web Services. O'Reilly.
L Seligman. (2002). Data Integration: Where Does the Time Go? *EEE Technical Committee on Data Engineering* .
Lew Tucker -CTO Cisco, S. K. (2010). *Grid vs. Cloud: What's the Difference?* Retrieved from http://www.serverwatch.com/news/article.php/3918126/Grid-vs-Cloud-Whats-the-Difference.htm
M Bichler, T. S. (2006). Capacity planning for virtualized servers. *Workshop on Information* .
M Johnson, R. R. (2002). Sketch Data Models, Relational Schema and Data Specifications. *Electronic Notes in Theoretical Computer Science - ENTCS , vol. 61* , 51-63.
M. Milburn. (2006). Metabolomics for Biomarker Discovery. *Drug Discovery 2006* .
MA Ott, G. V. (2006). Correcting ligands, metabolites, and pathways. *BMC Bioinformatics 2006, 7:517* .
Martina Schad, R. M. (2005). Metabolic profiling of laser microdissected vascular bundles of Arabidopsis thaliana. *Plant Methods 2005, 1:2 doi:10.1186/1746-4811-1-2* .
MassWorks. (2010). *Revolutionary software for MS calibration and formula ID.* MassWorks.
McShan, D. C. (2005). Towards Inference of a Biochemical Ontology From a Metabolic Database. *Comp Funct Genomics. 2005 Oct-Dec; 6(7-8): 398–406* .
Min Zhao, H. Q. (2011). PathLocdb: a comprehensive database for the subcellular localization of metabolic pathways and its application to multiple localization analysis. *BMC genomics, (2010)(11)* .
Mircea Gh Negoita, B. R. (2005). *Real world applications of computational intelligence.* Springer.
MP Papazolou, P. T. (2007). Service-Oriented Computing:State of the Art and Research Challenges. *IEEE Computer Society - 0018-9162/07* , 64-71.
MSDN, M. (2011). *Physical and Logical Databases*. Retrieved 2011, from http://msdn.microsoft.com/en-us/library/dd355372.aspx
Mueller, L. Z. (2003). AraCyc. A Biochemical Pathway Database for Arabidopsis. Plant Physiology. . *Plant Physiology. 132(2):453-60.*
Myerson, J. (2009). *Cloud computing versus grid computing*. Retrieved from IBM Developerworks: http://www.ibm.com/developerworks/web/library/wa-cloudgrid/

N. Gehlenborg, e. a. (2010). Visualization of omics data for systems. *Nature Methods VOL.7 NO.3s* .
NIH, N. (2012). *NCI's Comprehensive Cancer Database*. Retrieved from http://www.cancer.gov/cancertopics/pdq/cancerdatabase
O Fiehn. (2002). Metabolomics–the link between genotypes and . *Plant Mol. Biol. 48:155-171.*
O. Fiehn, L. W. (2007). Minimum reporting standards for plant biology context information in metabolomic studies. *Metabolomics Volume 3, Number 3 (2007), 195-201, DOI: 10.1007/s11306-007-0068-0* .
Paul Anderson, A. M. (2010). Cloud-based Map-Reduce Architecture for Nuclear Magnetic Resonance based Metabolomics. *Proceedings of the 7th Microsoft Research eScience Workshop* , 10.
Q Zhang, L. C. (2010). Cloud computing: state-of-the-art and research challenges. *Journal of Internet Services and Applications,Volume 1, Number 1 (2010), 7-18, DOI: 10.1007/s13174-010-0007-6* .
R de Knikker, Y. G.-l.-H. (2004). A web services choreography scenario for interoperating bioinformatics applications. *BMC Bioinformatics 2004, 5:25* .
R G. Shulman, D. L. (2005). *Metabolomics by in vivo NMR .* Wiley.
R L. Last, A. D.-H. (2007). Towards the plant metabolome and beyond. *Nature Reviews Molecular Cell Biology 8, 167-174* .
R Smyth, J. L. (1990). Early flower development in Arabidopsis. *Plant Cell. 1990 August; 2(8): 755–767.*
Rhee SY, Z. P. (2005). *AraCyc: Overview of an Arabidopsis Metabolism Database and Its Applications for Plant Research.* Springer. volume 57. pp. 141-153.
Rihn, B. (2003). From transcriptomics to bibliomics. *Med Sci Monit, 2003; 9(8): MT89-95* .
Roger, S. (2011). Efficient matching of multiple chemical subgraphs. *int-conf-chem-structures.* 9th ICCS.
S Kramer, L. D. (2001). Molecular feature mining in HIV data. *Proceeding KDD '01 Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining* .
S Weerawarana, F. C. (2005). chpt 3.3.1. In *Web Services Platform Architecture: SOAP, WSDL, WS-Policy,WS-Addressing, WS-BPEL, WS-Reliable Messaging, and More.* Prentic Hall, PTR.
Sakata, K. (2009). Soybean Proteome Database: A Data Resource for Plant Differential Omics. *J. Proteome Res., 2009, 8 (7), pp 3539–3548* .
Sauer, L. (2011). *PathLocDB - Pathway Localization database - parsing the flat file*. Retrieved from github-gist: https://gist.github.com/1471703
Sayle Roger. (n.d.). *Improved SMILES Substructure Searching*. Retrieved 08 2011, from Daylight Chemical Information Systems, Inc.: http://www.daylight.com/meetings/emug00/Sayle/substruct.html
Schmidt, C. W. (2006). Signs of the Times: Biomarkers in Perspective. *Environ Health Perspect. 2006 December; 114(12): A700–A705.*
Sencha. (2011). *ExtJS*. Retrieved from http://www.sencha.com/products/extjs/testimonials/

Șerban Mol-doveanu, V. D. (2005). *Sample preparation in chromatography theoretical aspects of the derivatizaton process.* Elsevier pg 479f., Chp 'theoretical aspects of the derivatizaton process'.
Swarbreck D, W. C.-H. (2008). The Arabidopsis Information Resource (TAIR): gene structure and functionannotation. *Nucleic Acids Res 36: D1009–D1014* .
T J Lee, Y. P.-C. (2006). BioWarehouse: a bioinformatics database warehouse toolkit. *BMC Bioinformatics 2006, 7:170* .
T Kind, O. F. (2010). Advances in structure elucidation of small molecules using mass spectrometry. *Bioanal Rev. 2010 December; 2(1-4): 23–60.*
T Yutaka, F. H. (2003). The Characterization and Metastable Decomposition of Isobaric Ions from Several Compounds. *J. Mass Spectrom. Soc. Jpn.Vol. 51, No. 2, 2003* .
TAIR. (2000). *Community Standards for Arabidopsis Genetics.* TAIR.
Tiago Grego, J. D. (2010). Chemical and Metabolic Pathway Semantic. *PLoS computational biology 6 (9), e1000937, 7, 2010* .
Tohsato Y, M. H. (2000). A multiple alignment algorithm for metabolic pathway analysis using enzyme hierarchy. *Proc Int Conf Intell Syst Mol Biol. 2000;8:376-83* .
TT Tanimoto. (1957). Similarity measures based on binary fingerprints. *IBM Internal Report 17th Nov* .
U Roessner, J. B. (2009). What is metabolomics all about? *BioTechniques 46:363-365* .
Ullman, J. (1983). *Principles of database systems.* Austrian Union Catalog.
V Silva. (2009). User Interface Concepts. In *Practical Eclipse Rich Client Platform Projects.* Springer.
W Cleveland. (1985). *The elements of graphing data.* Wadsworth Publ. Co.:.
W Weckwerth, O. F. (2002). Can we discover novel pathways using metabolomic analysis? *Current Opinion in Biotechnology Volume 13, Issue 2, 1 April 2002, Pages 156–160* .
W. Weckwerth, K. M. (2005). Metabolomics: from pattern recognition to biological interpretation. *Drug Discovery Today, Volume 10, Issue 22* , 1551–1558.
W3C, H. H. (2004). *W3C Web Services Glossary*. Retrieved from http://www.w3.org/TR/ws-gloss/
Weckwerth W, B. S. (2008). The multinational Arabidopsis steering subcommittee for proteomics assembles the largest proteome database resource for plant systems biology. *J Proteome Res 7: 4209–4210* .
Weckwerth, W. (2011). Green systems biology. *J Proteomics. 2011 Dec 10;75(1):284-305* , 22.
Weckwerth, W. (2003). METABOLOMICS IN SYSTEMS BIOLOGY. *Annu. Rev. Plant Biol. 2003. 54:669–89* .
Wolfram, W. (2007). Metabolomics: methods and protocols. In W. Wolfram. Humana Press.
Wuster, A. (2008). Chemogenomics and biotechnology. *Trends in BiotechnologyVol.26 No.5* .
Z Wu. (2009). Research on Distributed Architecture Based on SOA. *Communication Software and Networks, 2009. ICCSN '09. International Conference on.* IEEE.
Zhang, P. F. (2005). MetaCyc and AraCyc. Metabolic Pathway Databases for Plant Research. *Plant Physiology 138: 27-37.*

8. CV - Curriculum vitae

Personal history:



Name:	Lorenz Sauer
Email:	lorenz.sauer (AT) gmail.com
	lorenz.sauer (AT) univie.ac.at
Birthplace:	Wien, Austria
Citizenship:	Austrian

Education:

’90	Elementary School
’94	Bundes Real Gymnasium f. Math. (sec. school for Mathematics; grad. w. distinction)
‘98	HTL (Polytechnic school[44]: Electronics & Technical Informatics; Thesis: Artificial Vision Interface - AVI)
’04	TU Wien, Physics
‘05	UNI Wien, Molecular Biology
’06	UNI Wien, Chemistry
’12	(projected) Diploma in Molecular Biology w. distinction[2]
’12	(projected) M.Sc. in Biological Chemistry

Employment / work experience:

’99	• Work Trainee , Gesig (60day practical, as part of the polytechnic curriculum) • Computer Assistant, Federal Austrian Forest Research Centre, Dep. of Genetics and Silviculture (Supervisor: Dipl. Ing. I. Strohschneider)
‘00	• Technical Assistant (examining, fixing of Printed Circuit Boards), Gesig, Supervisor: Ing. K. Weingartner
‘01	• Freelance Programmer at Microweave, project for : Dipl.- Ing. Dr. F. Müller
‘02	• Backend programming, WebTech, (Supervisor: T. Prieler)
‘03	• Design & Programming WunderWerk,(Supervisor: Mag.(FH) B. Sendlhofer) • Network Programming, Multiart, (Supervisor: S. Auferbauer) • GSM-Backend -Programming: Mobile Phone Service Platform (Mag. R. Tie fenbacher)
‘04 -‘09	•short-term intern or freelance programming positions, including FPGA pro-

[44] practical experience in hours according to the diploma: Workshop: 1200h / Laboratory: 1000h ; 60d internship

gramming

’11 • University of Vienna, MoSys – Diploma-Student

Languages:

• German
• Spanish (fundamentals)
• English

Misc:

• Oxford University First Certificate in English ‘03
• CISCO Networking Certification ‘04
• Summer School on Molecular Modeling and Drug Design ’08 (Yeditepe Üniversitesi)
• Late Summer Practical Proteomics (IMBA) ‘11

Laboratory Skills:

• Microbiology, Molecular biology and Biochemistry standard- and advanced procedures
• NMR (practical work on Osteopontin), *Software*: Sparky; peak assignment of 3D HSQC protein spectra
• Crystallography (practical work on Chloride Dismutase), Xtallography -*Software*: Coot;
• full protein 3DHSQC NMR workup: custom cDNA vector, transfection, mutant protein over expression, preparative protein purification, (spin-)labeling, NMR measurements, peak assignment, folding w. restraint parameters, modeling
• analytical separation science, in particular various hyphenated system to NMR and MS and systematic protocol development
• Primer design, analysis and implementation of various amplification methods
• Bioinformatics since ’10, including annotation of prokaryotic Genomes

Safety:

• Working in workshops, and electronic/telecommunication laboratories (till ‘05)
• Working in physical laboratories (’05), biological/chemical laboratories (current)
• Knowledgeable in laboratory safety regarding the correct handling of chemical and hazardous biological materials.
• Knowledgeable in handling sensitive electronics devices
• Trained in toxicology and in providing first aid

Sofware Skills:

- Frontend & Backend - Design/Programming
- Networking & Routing (CISCO Networking Certification)
- Programming (hardware level: assembler/disassembler, various uProc's, ANSI C, OOP code w. interface definitions; overall experience: > 200.000 lines of deployed programming-code)
- Parser and assembler design (state machines)
- PCB Design & Routing to various EMS (Electro Magnetic Susceptibility) standards,
- PCB Fabrication and pre/post workup
- Electronic Simulation Suites (Orcad 9.x, UltraSim 2001, Protel 99SE, Eagle 4.x)
- CAM & CAD / 3D Design (Solidworks, 3DSMax 6.x, Maya 5.x, Cinema 4D XL 8.x)
- Graphics / UX design
- visual/image processing & image tracking – theory and implementation (HTL thesis)
- Extensive knowledge in Win32-APIs; implementation of a Windows Application Framework

- ***Programming:*** ANSI C , C++ , C# , ASM {x86, PIC16F28} , JAVA , PHP , PYTHON , JavaScript
- ***Database:*** ANSI SQL (inc. variations of Mysql & PostgreSQL) , SQL 97, XQL, LINQ
- ***Markup/Meta Languages:*** XML , XSL/XTSL , XPATH, CSS , XHTML , WML , WSDL & SOAP, HTML5

- Matlab (Cheminformatics, Bioinformatics –Application of Artifical neural networks for spectral separation through identification ; practical '09); since '10 R-language
- Mathematica, Mathcad
- Molecular modeling (Chemdraw 3D, VMD; GAMESS for Mol. Dynamics)

→ *a reference list of IT projects or laboratory work, is available upon request*

"Anyone who has lost track of time when using a computer knows the propensity to dream, the urge to make dreams come true and the tendency to miss lunch."

Tim Berners-Lee ('inventor' of the Web)