**Versioning and contribution history**

|  |  |  |
| --- | --- | --- |
| **Version** | **Description** | **Contributors** |
| 0.1 | First draft with Introductin, soe copy-and-paste, outline | Frithjof Dau, Kenneth McLeod |
| 0.2 | Work on “key messages” | Frithjof Dau |
| 0.3 | Work on “semantic search” | Frithjof Dau |
| 0.4 | „semantic search“ and „architecture“ |  |
| 0.5 |  |  |
| 0.6 |  |  |
| 0.7 |  |  |
| 0.8 |  |  |
| 1.0 |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Contents

[Introduction 2](#_Toc413157481)

[Project Setup and Key Technologies 5](#_Toc413157482)

[Introduction into CUBIST 5](#_Toc413157483)

[Project Setup and Use Cases 6](#_Toc413157484)

[Introduction into Semantic Technologies 6](#_Toc413157485)

[Introduction into Formal Concept Analysis 6](#_Toc413157486)

[Key Messages and Theses 8](#_Toc413157487)

[CUBIST Prototype 11](#_Toc413157488)

[Architecture 11](#_Toc413157489)

[Different Means to Access Information 13](#_Toc413157490)

[Semantic Search and Query Generation 14](#_Toc413157491)

[Explorative Search 16](#_Toc413157492)

[Conceptual Scaling 16](#_Toc413157493)

[Visual Analytics 16](#_Toc413157494)

[Outcome 18](#_Toc413157495)

[User Evaluation 18](#_Toc413157496)

[Our Take 19](#_Toc413157497)

[Conclusions 21](#_Toc413157498)

[Bibliography 22](#_Toc413157499)

CUBIST: Combining and Uniting Best Investigators in a Super Team

Frithjof Dau (SAP SE), Ken McLedo (HWU)), and lll

Abstract: lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum

# Introduction

The constantly growing amounts of data and an emerging trend of incorporating unstructured data into analytics is bringing new challenges to Business Intelligence (BI). Traditional Business Intelligence (BI) solutions fall short in different aspects. Firstly, they focus only on structured data and disregard the increasing amount of information hidden in unstructured data. Secondly, they focus solely on quantitative analysis of the data: qualitative aspects, like structural dependencies in the data, are neglected.

In the EU funded research project CUBIST, which ran from October 2010 to September 2013, those two drawbacks have ben adressed. CUBIST’s approach was to complement traditional BI solutions. he CUBIST project developed methodologies and a platform that combines essential features of Semantic Technologies and BI. The most-prominent deviations from traditional BI-platforms are:

* The data persistency layer in the CUBIST-prototype based on a BI enabled triple store, thus CUBIST enables a user to perform BI operations over semantic data.
* In addition to some traditional charts, CUBIST provides novel graph-based visualizations to analyse the data. As mathematical foundation for meaningfully clustering the data, Formal Concept Analysis is used.

From a user’s perspective, CUBIST provides three different means to access the data:

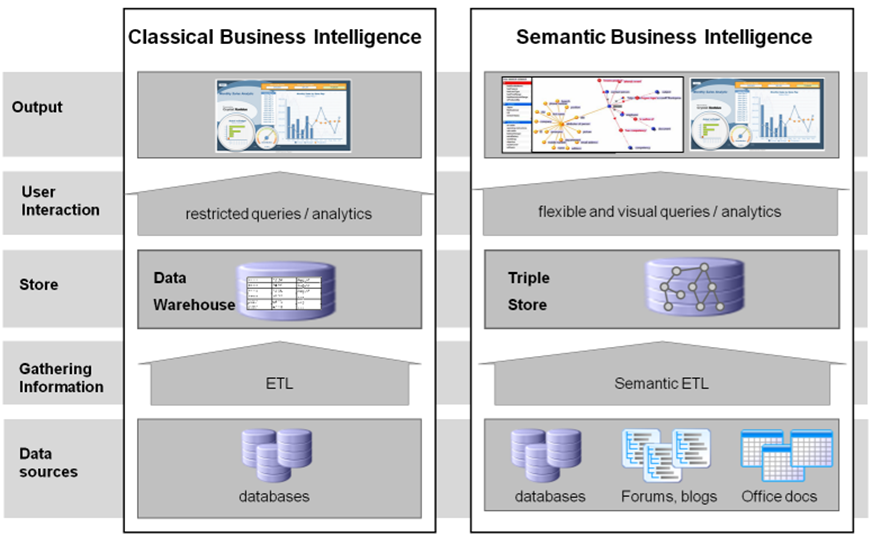
* Factual search: First of all, using a semantic search allows the user to specifically the query the data, in order to retrieve small or large sets of entities which satisfying user-defined constraints.
* Explorative search: A graph-based view allows the user to interactively explore the data.
* Visual analytics: Clusters and aggregations of data can be visually analyzed using traditional charts or novel visualizations. The selection of the visualized data as well as the visualizations are highly interactive, thus CUBIST provides “BI as a self-service”.

A first, general diagram which depict the differences between traditional BI and CIBUST is depicted below in Figure whatever.

CUBIST was the joint effort of seven partners, namely the project coordinator SAP SE (SAP AG during the project runtime, SAP, Germany), Ontotext (ONTO, Bulgaria), Sheffield Hallam University (SHU, UK), Centrale Rechereche S.A. (CRSA, France), Heriot-Watt University (HWU, UK), Space Applications Services (SAS, Belgium), Innovantage (INN, UK).

SAP, ONTO, SHU and CRSA served as technological partners. During the project, they have developped a web-based prototype with the following capabilities:

* Unstructured and structured data is federated in a triple store.
* The prototype supports diferent information needs, i.e. searching for data, exploring and navigating in the data, and analysing the data.
* The protype supports “BI as a selfservice” via comprehensive filter mechanisms and leveraging the semantics of the data.
* The Visual Analytics of CUBIST consists of an extensive set of novel and highly interactive diagrams which allow both the quantitative and qualitative analysis of the data.

****

Use cases have been provided by the use case partners HWU, SAS, and INN. They have provided requirements and data sets at the beginning and constant feedback during the project lifespan, until they have finally evaulated the prototype at the project end. In CUBIST, HWU explored the semantic representation of biomedical data from a biomedical atlas and a gene expression database. SAS investigated telemetry data of a specific research device in space they monitor in mission control rooms. Finally, INN provided a job market use case which combined information from job advertisements crawled by CUBIST and an existing firmographic database.

This paper describes the approach of CUBIST, the protoype which has been developped, and the results oft he project –particulary the evaluation in detail. The project as such, particularly its setup, the use cases, core technologies, and key theses are provided in section XXX. The prototype is described both in terms of technology and architecture as well as in terms af capabilites is described in section XXX: The results oft he project, particulary the results of the evaluation, are described in Section XXX.

# Project Setup and Key Technologies

## Introduction into CUBIST

Research Challenges and Expected Impact

CUBIST is based on Semantic Technologies, particularly on RDF for data representation and triple stores for federating and persisting the data, and Formal Concept Analysis (FCA) for conceptually clustering the data and organizing the clusters into hierarchical relationships. These clusters will be used for the Visual Analytics. This leads to the following challenges and expected outcomes:

Semantic Extract, Transform and Load (ETL): Source data in unstructured sources might be noisy, inconsistent and incomplete. Extracted data from unstructured sources has to be brought into relationship with extracted data from structured sources. CUBIST targets semantically enriched lineage information, error detection and identity resolution within extracted data, and a semantic ETL component that provides SPARQL (the query language for triple stores) endpoints for various data sources.

Query Language: SPARQL lacks complex aggregate functionality, reporting functions and rollup/cube expressivity. In alignment with the efforts of W3C, CUBIST will extend SPARQL by needed OLAP functionalities.

Performance and scalability of the Triple Store: Current state-of-the-art implementations of triple stores are for tens of billions of triples, a magnitude lower than the data volumes for a state-of-the-art data warehouse. CUBIST will significantly multiply the number of triples the triple store can deal with.

FCA and Triple Stores: Most FCA applications are stand-alone solutions. In CUBIST, a layer within the warehouse will integrate the triple store with the FCA-based visual analytics.

Scalability of FCA: Existing FCA solutions do not scale to large amounts of data. CUBIST will investigate high-performance FCA algorithms and tools, including parallel processing algorithms for multi-core architectures.

Visual Analytics: Current FCA visualization tools have been designed without very large data sets in mind. Interlinked with best practices from known BI visualizations, CUBIST will scrutinize novel approaches for FCA-based visualisations which allow for in depicting, navigating through and visually querying the data.

To summarize, for the core fields Semantic Technologies, Business Intelligence and Visual Analytics, we expect the following impact:

Semantic Technologies: CUBIST aims to bring Semantic Technologies to a level where they can be successfully applied in industrial settings using huge data sets, comparable to established technologies such as relational databases and BI.

Business Intelligence: It is expected that incorporating unstructured data will be very important for future BI systems. CUBIST takes an essential step in this direction. From a technological perspective, CUBIST will have an impact on the architecture of future BI systems. From a business perspective, CUBIST will help overcome the barrier of complexity of BI tools and interfaces and apply BI functionalities to new business scenarios and new user groups.

Visual Analytics: From a technological perspective, using FCA for representing and navigating the large amounts of data will open up FCA to new horizons in terms of applicability in real business settings. From a user perspective, using FCA has a solid mathematical foundation and a close link to the human perception of concepts, thus FCA will drive Visual Analytics to a new level of theoretically precise and humanly comprehensible visualizations.

## Project Setup and Use Cases

Lorem ipsum blablaa

## Introduction into Semantic Technologies

1. Good: schema last (CUBIST would not work with RBDMS)
2. Good: Graph-based schema good for graph exploration, which is appreciated (see V.A.)
3. TS: graph db is suited for \*specific\* use cases.
4. **Beyond SoA for ST**
5. bad: performance
   1. TS is essentially \*transactional\* repository.
   2. in RDBMS, there was until recently a distinction between operational and transactional prevalent
      1. (changed recently with in\_Memory, like HANA, MS Hekaton)

Lorem ipsum blablaa

## Introduction into Formal Concept Analysis

Formal Concept Analysis (FCA) is a Semantic Technology designed to mine relationships within data. FCA was introduced by Rudolf Wille in 1984 and builds on “applied lattice and order theory that was developed by Birkhoff and others in the 1930’s" [wille]. It was initially developed as a subsection of Applied Mathematics based on the mathematisation of concepts and concepts hierarchy. A formal context is a triple (G,M,I) consisting of a set of objects, G, a set of object attributes, M, and a relation I between G and M. A formal concept (A,B) is a set of objects, A⊆G, and a set of attributes, B⊆M, such that all objects in A have all the attributes in B, and there are no other objects in A that have all attributes in B and no other attributes in B that are shared by all objects in A. More formally, if a pair of closure operators are defined by:

Then a formal concept, (A,B), is defined as and . B is called the intent of the concept and A is called the extent. A formal concept is, therefore, a closed set of object/attribute relations, in that its extension contains all objects that have the attributes in its intension, and the intension contains all attributes shared by the objects in its extension. Table ? shows an example of a formal context of genes and EMAP (embryonic mouse) tissues in which the genes were detected. The rectangle of bold crosses represents a formal concept containing genes Upk2 and Itpr3.

Table 1: A simple context of genes (objects) and tissues (attributes) from the EMAP anatomy. The bold crosses indicate the formal concept {{*Upk2* and *Itpr3*}, {Turbinate bone TS23, Tibia TS23 and Fibia TS23}}.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 3\* | Turbinate |  |  |  | Orbito- |  |
|  | bone | Tibia | Fibia | Femur | sphenoid | Humerus |
|  | TS23 | TS23 | TS23 | TS23 | TS23 | TS23 |
| *Upk2* | × | × | × |  |  |  |
| *Itpr3* | × | × | × | × |  | × |
| *Cops7b* |  |  | × | × |  |  |
| *Tgfbi* | × |  |  |  | × |  |

A concept lattice visualisation (also known as a Hasse diagram) arises from the hierarchical relation between concepts [priss]: when an attribute is added to intent the corresponding extent becomes smaller (more specialised) and vice-versa.

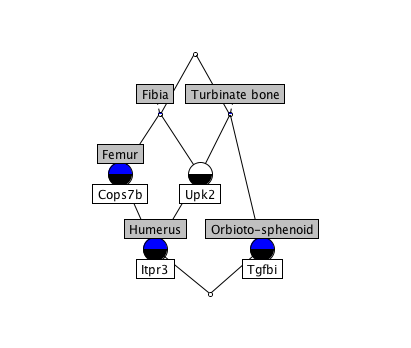


Figure 1: Lattice visualisation of the formal concepts detected in the context from Table ?. Attributes have grey labels above the nodes, and the objects white labels below the nodes.

Each node is a concept, where the intent contains all attributes at and above the node (genes in this case) and the extent contains all the objects at and below the node. To read the intent for the bottom left node (Humerus / Itpr3) in Figure ? a user must read all the attributes contained on a path between the current node and the top node. The paths are:

Itpr3 / Humerus > Cops7b / Femur > Fibia > Top

Itpr3 / Humerus >Upk2 / Tibia > Fibia > Top

Itpr3 / Humerus > Upk2 / Tibia > Turbinate bone > Top

Accordingly, this tells us the object (Itpr3) is expressed in: Humerus, Femur, Fibia, Tibia and Turbinate bone. Furthermore, where ever the genes Cops7b and Upk2 are expressed, Itpr3 is also expressed.

To read the extent for the node labelled “Turbinate bone" (top right) a user must read all the objects contained on a path between the node of interest and the bottom (lowest) node. This informs the reader that the turbinate bone has the following genes expressed in it: Upk2, Itpr3 and Tgfbi.

Because FCA is predicated on binary relations (an object either has an attribute or not), traditional, many valued, data have to be converted in CUBIST to this Boolean form, a process in FCA known as scaling [wolff, shu\s\do5(1)0]. A data item with n possible values becomes n formal attributes. An original attribute for Gender, for example, with the possible values Male and Female, is scaled in FCA as two formal attributes, Gender-Male and Gender-Female. Each tissue in Table ?, for example, is a formal attribute. A continuous, numerical, item can be discretised by using numerical ranges. In CUBIST, an existing tool called FcaBedrock [shu\s\do5(1)0] was further developed, adapted and integrated into the system for this purpose.

Applying FCA to data sets is an intensive process, computationally speaking. Computing formal concepts is an exponential task, with often hundreds of thousands, if not millions, of concepts being present in a data set. To deal with this, an existing high-performance concept-mining algorithm called In-Close [simon\s\do5(0)9] was implemented in the CUBIST system. When the concept miner is combined with a query (that produces a sub-set of the data), CUBIST is an efficient system with fast response times.

## Key Messages and Theses

Traditional BI tools allow the user to analyze the user along different dimensions, and provide diagrams like bar charts, pie charts, scatter plots, line dagrams, etc. All activities are acting on the (statistical) analysis of \*numerical\* data, data which is understood to be in some form of a table structure. [Citations needed? Point to some BI vendors, even AP?]

More recently, e.g. in the field of big data and social nets, vast amounts of data is model not to a tabular, but to a graph-based model. On the storage and retrieval side, graph-databases are used. On the user search side, graph-based search are used (e.g. “graph search” from facebook) and on the analytical side, graph-based visualizations are used which depict entities, relations, and clusters in the data (e.g. lnMaps from LinkedIn).

Cubist follows to some extent a specific graph-based approach: On the storage and retrieval level, we use a Triple Store. On the search side, a specific semantic search is used, and on the side of Visual Analytics, Formal Concept Analysis is used as main means to analyze the data.

In our understanding, traditional BI tools provide *quantitative* analysis means, and graph-based, particularly FCA-based, tools provide *qualitative* analysis means. This shall be exemplified with a small toy example. We consider the following dataset:

|  |  |
| --- | --- |
| Skill | Persons with that Skill |
| IE | Anja, Ben, Ernst, Fred, Ken |
| ETL | Chris, Fred, Mark |
| BI | Ben, Chris, Fred, Lemmy, Mark, Naomi |
| ST | Anja, Diana, Ernst, Fred, Gerald, Harriet, Ken, Owen |
| FCA | Anja, Diana, Gerald, Harriet, Ian, John, Ken, Owen |
| VIZ | Anja, Diana, Ian |

This dataset can be confronted with different information needs. Examples are:

1. Show me the count of people for a given skill
2. Show me the skills and how how strongly they are connected.
3. Show me the skills and people such that I get an idea of the distribution of skills among people, and dependencies between skills.

Obviously, 1. Is a traditional BI need, whereas 2. has a graph-based flavor, and 3. Is a need which fits into the FCA-approach. We aim at using some appropriate visual analytics means of the respective field, and we will answer the information needs with the following means:

* A bar chart, being a traditional BI Visual means, to address need 1.
* A force-based layout, being a graph-based visualization, to address 2, and
* A concept lattice diagram, based on FCA; to address need 3.

In order to apply the respective analysis means, the data has first to be converted into an appropriate form. This is depicted next. The table on the left, is obtained via counting the number of people per skill, and the table in the middle via counting the number of people who sshare two skills.

|  |  |  |
| --- | --- | --- |
|  |  |  |

These datasets can be now visualized as follows:

|  |  |  |
| --- | --- | --- |
|  |  |  |

Charts like bar charts, pie charts, ets are very common and qell-known visualizations, They are easy to understand, and easy to (automatically) layout. They are a perfect tool to analyze numbers. On the other hand, they are (deliberately) not offering all information of the intial dataset. In our example, we loose all information about the people. Attributes may overlap, as in this example, where people are counted manifold, thus overlapping attributes are badly represented. Relationships betwwen entities, finally, are not utilized at all.

Graph-based means can provide attractive visualizations, which are (relatively) intuitive and easy to understand. In contrast to the traditional means, they are goot at utilizing relationships in the dats, but bad for analyting the numerical facets of the data.

Finally, in the FCA-visualization, we have no loss of information, Meaningful clusters in the data are depicted, and the graph shows dependencies between entities (in our example, for both people and skills), On the other hand, similar to the other graph-based visualizations, it is not suited to analyse numbers. Moreover, as the number of nodes might be magnitudes higher than the origina data, the approach does not scale. For lattices, finding a good layout is on general not easy and can hardly be automated. Finally, FCA is obviously a unfamiliar means to analyze data.

This short discussion shows that each visualization has ist own strengths and weaknesses, Please note that the three visualizations are perfectly suited to address information needs 1., 2., and 3., in that order. More generally, in our understanding *each type of visualization is suited for a specific type of information need*. Thus the different kinds of visualizations are not competing, but complementing. As for example demonstrated by the use cases in CUBIST, the information needs ehich occur when data is analyzed data go beyond a purely numerical view. Thus our main thesis is: Future BI tools need to address more kinds of information needs which go beyond the numerical analysis of data, thus they should provide different types of visualizations and analysis gmeans (e.g. traditional as well as graph-based visualizations and analysis means), for example side by side with linking-and-brushing approach.

# CUBIST Prototype

## Architecture

In this section, the architecture of the CUBIST prototype is described in two levels of granularity. Figure 1 depicts the *reference architecture*. This is an architecture based on *generic, abstract* components. For example, as persistency layer, this architecture names the Semantic Data Warehouse, being a database for RDF-data (i.e., a triple store). There are several triple stores available, e.g. OWLIM from Ontotext or Allegro Graph from Franz Inc. The Reference Architecture outlines the various layers and components comprising a CUBIST system, as well as the interactions and data flow which exists between these components. It is out of the scope for the Reference Architecture to prescribe low level functionalities, APIs, message exchange patterns between components, or specifics of the various data sources. In any concrete implementation of this architecture, a concrete triple store has to be chosen. The same considerations apply to the other components in the reference architecture as well. The CUBIST prototype is one (of many) possible implementations of this architecture where each generic component of the reference architecture is replaced by one (or sometimbes many, e.g. for the SETL-component) concrete components. The implementation architecture of the CUBIST prototype depicted in Figure 2, which shows the implementation specification and describes the concrete details by referring to the names of the software products – both CUBIST specific and 3rd party – that are used by the prototype.

The reference architecture defined in the CUBIST project consists of five layers:

* The *Data Layer* includes all structured and unstructured data sources relevant to the CUBIST system. Examples for such data are: structured relational databases and Excel files, unstructured web documents, semi-structured XML documents.
* The *Semantic ETL Layer* comprises different software products and components that help accessing, extracting and transforming data into a unified RDF data model so that the legacy data can be stored in the RDF data warehouse of the CUBIST system.
* The *Semantic Data Warehouse* is a persistence layer containing a high-performance RDF database, responsible for storing and querying of semantically enriched data and its related schema information (ontology). In addition, this layer contains a Data Management API which enabled the semantic ETL tools to store and update data in the RDF data warehouse; as well as a SPARQL endpoint providing a data query and access interface.
* The *CUBIST Services Layer* includes the Search Service component (which offers application logic to navigate and search the semantic data warehouse) and the FCA Service component (which manages the *formal context* lifecycle).
* Finally,the *CUBIST Frontend Layer* is comprised of all GUI tools used by the end users. They connect to the underlying CUBIST services layer and visualize information coming from the semantic data warehouse.

Next we describe how this reference architecture has been instantiated in the CUBIST prototype. The datasources have been already described in the use case descriptions. For the HWU and SAS use case, only structured data had to be transformed and loaded into the information warehouse. CUBIST followed a standard ETL-approach using *Talend Open Studio[[1]](#footnote-1)* which is an comprehensive open source software with an easy-to-use graphical development environment for data integration. It The RDF extensions make the whole machinery applicable to semantic data processing as well. Moreover, in the third use-case, more tools like *D2R Server[[2]](#footnote-2)* for loading data from a relational database into the triple store, or GATE for information extraction from text and html-documents have been used.

For modelling the respective ontologies, the free version of TopBraid Composer has been used.

As all components for data integration and ontology modelling are 3rd party components, we do not go into more details, but focus on the levels where the data integration already happened.

The Semantic Data Warehouse has been in CUBIST instantiated by **BIG OWLIM v5.3,** beinga highly scalable triple store developed by Ontotext. It is implemented in Java, it is Sesame API compliant and supports RDFS and specific OWL profiles. The warehouse is available to the services layer components via a standard SPARQL endpoint (accessible through a Java API or RESTful interfaces).

**NowaSearch Front-end and Search Service (SAP)** is a web-based research prototype for semantic information integration and search with faceted search features. It is the outcome of a former research project at SAP (Aletheia) and adapted for CUBIST. This prototype enables a factual search (based on keywords or based on semantically enriched information), faceted search and graph exploration for the information stored in the semantic layer. It serves now as the basis for the CUBIST integrated prototype: other components are integrated into this prototype. NowaSearch is adapted for CUBIST in order to meet the information access needs of a BI application.

**FCAService (SHU)** is a component developed by SHU that provides Web Services for converting results of SPARQL-queries into formal contexts. This is done via clustering the values of an attribute into so-called “bins”. For different attribute types, it allows a fine-grained tunable conversion: It provides different methods on computing or manually entering the borders of the bins, or it can be set whether bins are disjoint or whether they can include each other.

FCAService has been published on GitHub under the Apache 2.0 license as open source.

**CUBIX (CRSA)** is a standalone, frontend based FCA visualization / analysis tool. The tool is being developed in CUBIST from scratch and continuously refined with the actively involvement of our three users groups and their use cases. A first version of the tool, which has been developed in the starting phase of CUBIST, has been based on flash and the flare visualization library. During the course of project it became evident that flash becomes an obsolete technology and will be superseded by HTML5, CUBIX has been redeveloped in HTML5, mainly using d3.js as visualization library.

CUBIX already provides novel features for visualizing large concept lattices (e.g. the transformation of lattices into trees) and implements the gathered visualisation requirements. Typical uses of CUBIX include semantic data analysis and pattern detection, anomaly detection, comparisons, information classification, and knowledge discovery. As of the end of the second year of CUBIST, CUBIX provides numerous visualizations for the lattices (Hasse diagrams, sunburst diagrams, trees, treemaps, icicles, scatter plots) as well as bar charts which show the distribution of analysed entities, as well as searching, selecting and filtering capabilities for the visualizations.

CUBIX has been published on GitHub under the Apache 2.0 license as open source.

|  |
| --- |
|  |
| ImplementationArchitecture |

Figure 1: FMC diagrams1 FMC diagram of the CUBIST reference architecture

## Different Means to Access Information

From a user’s perspective, CUBIST provides three different means to access the data:

* Factual search: First of all, using a semantic search allows the user to specifically the query the data, in order to retrieve small or large sets of entities which satisfying user-defined constraints.
* Explorative search: A graph-based view allows the user to interactively explore the data.
* Visual analytics: Clusters and aggregations of data can be visually analyzed using traditional charts or novel visualizations. The selection of the visualized data as well as the visualizations are highly interactive, thus CUBIST provides “BI as a self-service”.

The general prototype has been tailored to three use cases, provided by HWU, SAS and INN. Those partners provided datasets, requirements and ongoing feedback. Finally, the prototype has been evaluated by these partners.

## Semantic Search and Query Generation

To search for entities, CUBIST provides a faceted search frontend with additional navigation features. The corresponding tab is titled “Search and Select”. This shall indicate that this tab can be on the one hand used “standalone” to perform a factual search and investigate details of the found entities, and on the other hand, it serves as starting point to select data for the graph exploration and analytics means of CUBIST, i.e. it is ie the entry point for all other activities. Business users using CUBIST for information retrieval thus may browse and later extract needed information (cp. [Ellis1989, Ellis1989a]) using this frontend. The reference component supports for faceted navigation through the underlying semantic graph data base. It also offers filtering and search to allow for formal and informal search (cp. [Choo1998]).

The conceptual design of the UI and the envisioned workflows is tailored for searching, exploring and analyzing a “well-formed” (this term will be explained later) RDFS-ontology. In a BI understanding, the schema of this ontology serves as “semantic layer” for the data. The term “semantic layer” has been developed and patented by Business Objects (meanwhile being part of SAP SE) and refers to a user-friendly terminology of (business) data, thus is it metadata for the data. Essentially, a semantic layer consists of

* hierarchies of elements (in standard BI applications like “region”, “product”, “time”
* attributes (like “name”, “size” etc)
* measures (numerical, ie.e. numbers like “units sold” to be analyized in BI applications)

In a BI-application, the underlying data source, including their technical names etc, are mapped to the semantic layer, thus it shields the user form the underlying implementation of data storage. Obviously, there is a strong relation-ship between RDFS-schemata and semantic layers: Measures and attributes correspond to (data) properties in RDFS, and hierarchies in semantic layers can be easily modelled in RDFS. In CUBIST, we extensively utilize object properties as well, which have no direct counterpart in semantic layers. Anyhow, using an RDFS schema as semantic layer imposes several restrictions to the schema. First of all, we assume in CUBIST that for each property its domain and range are set with a rdfs:domain and rdf:range statement. Secondly, actually in no CUBIST use case, a hierarchies (of classes or properties) have not been needed, and the UI provides no direct means to support hierarchies. On the other hand, CUBIST allows multiple to assign values with on property to one entity (this is for example used for the synonumy is genes in the HWU use case). An RDF-schema which satisfy these constraints can serve both as semantic layer and as technical description of the data in the ontology, thus a big advantage of this approach is that no mapping from the terms in the UI to the technical data is needed anymore, which in turn is a key enabler for CUBIST to provide BI as a self-service.

In the “Search and Select”-tab, each class of the ontology (its name) is displayed, and all data properties having the class as domain are displayed as facet of the class. A textual searchbox below the class name allows the filter the facets. For CUBIST, this feature was particularly handy for the SAS-usecase, where more than 200 facets belong to the main class.

As each facet is the range of an RDF data property, a facet is either of type string, date, or number. CUBIST provides for the types different types for searching and selecting them. For strings, entities can be one by one selected (or deselected) with a checkbox. For numbers and dates, no single entities are selected: Instead, the user can specify one or more intervals by selecting their min and max in the UI. For all types, a search box below the facet allows to search directly for attribute values.

By setting filters in the facets, the user reduces the result set accordingly. Each entity belongs to one class in the ontology. The user has to select one of the classes as overall “domain of interest” via a radio-button in the ui. The resultset is restricted to members of that class. A distinguishing feature of CUBIST is that filters can be set across those classes, i.e. it is possible to use facets of on class to filter down the result set of the selected class. The algorithm behind will be explained later in this Section.

For the resultset, there are two ways, to be displayed, being represented as two tabs to the right of the panel with tha facets. The “Instances”-tab shows a list of the names of the found entities. Each entity can be unfolded, and then for all data- and object-properties, all values are displayed. The object-properties refer to other entities: Thus their values are clickable, and then for the new entity its details are shown (as now the user is not in the modus of displaying the resultset anymore, the view is changed from “Search and Select” to a so-called “Instance View”). Note that this feature allows the user to navigate through the whole information space along the object properties, thus this is an explorative feature of CUBIST.

In contrast to the “Instances”-tab, the “Datatable”-tab shows the result-set in a way not restricted to attributes of the domain of interest. Each attribute (whether it belongs to the domain of interest or not) can be selected to be an “attribute of interest” with a checkbox to its right. In the datatable-tab, the name of each entity and the selected attributes of interest are displayed in a tabular view. (in case one or more attributes of interest have multiple values, then multiple rows for the entites are displayed). The attributes of interest are moreover those attributes which can be analyzed futher by the visual analytics means of CUBIST. Again, it is a distinguishing feature that the selection of attributes of interest works across the RDF classes.

The overall query to the CUBIST repository is composed of the choice of the universe of interest, the aatributes of interest, and filter settings in the attributes. Those informations are stored as part of of the URL, thus the query can be easily stored via bookmarking the current CUBIST URL.

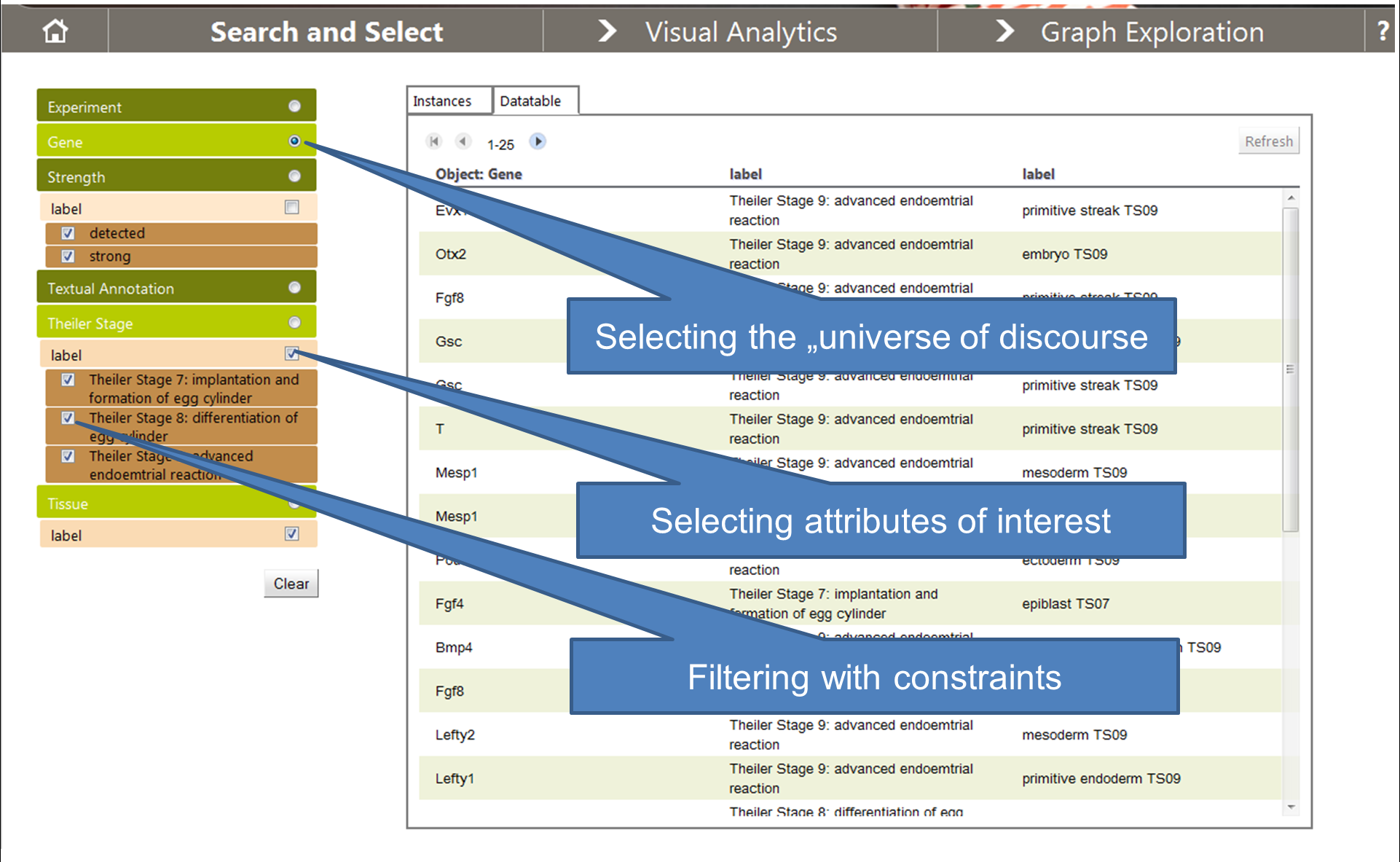


Fig 6: Searching and Filtering in the CUBIST "Search and Select" panel

Finally, it shall now be described how CUBIST deals with filtering or selecting attributes of interest across the classes in the ontology. This shall be exemplified with the user-query, as it can be seen in Fig 6: “Searching and Filtering in the CUBIST "Search and Select" panel. From the user-query, as defined in the CUBIST-frontned, a corresponding SPARQL-query is generated. The SPARQL-query essentially consists of a graph-pattern to be matched, SPARQL filter conditions to be met, and query variables used for the resultset as follows:

1. In the first step, all classes which are needed for the query are collected. The set of these classes comprise the universe of discourse, all classes of attributes of interest, and all classes where filters are set. In our example, this set contains Gene, Strength, Tissue and Theiler\_Stage.
2. Next, a minimal set of classes containing the classes of step 1 is constructed such that all classes in this set are connected (direct or indirect) via object-properties. In our example, Textual\_Annotation will be added to the four classes of step 1. We construct the subgraph of the ontology with the classes of step 1 and 2 and all object properties between those classes, apart from one (semantically irrelevant) restriction: CUBIST allows to use owl:inverseOf to model that object properties are mutually inverse, and in this case, CUBIST takes only one of those properties into account. The so constructed graph is the graph pattern of the SPARQL query. Note that in the CUBST ontologies, that subgraph is uniquely determined.
3. The name-attribute of the universe of discourse and the attributes of interest as chosen as query variables.
4. Filter conditions set in the UI are translated accordingly to filter conditions of the SPARQL query. Of course, multiple filter conditions for one facet are combined with a disjunction, and all those disjunctions for the different facets which are used for filtering are combined with a conjunction.

In our example, the naïve reading of the query is as follows: Retrieve all genes which are detected or even strongly detected in some tissue or Theiler Stage 7,8, or 9, and give me the names of the genes, the tissues where they are detected, and the Theiler stages of the tissues.

## Explorative Search

Lorem ipsum blablaa

## Conceptual Scaling

Lorem ipsum blablaa

The Context Creator fits in the overall architecture by converting the data into formal contexts suitable for FCA, hence making lattice visualization possible, in the sense that its output becomes an input to the High Performance Concept Miner, which, in turn, produces the output to be further processed by the Visual Analytics component:

Context File

High Performance Concept Miner

Visual Analytics

Context File

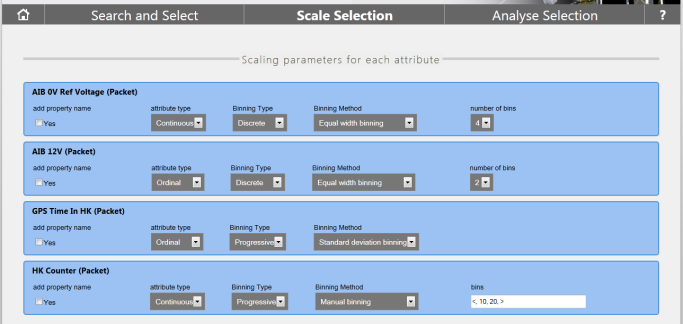
Context Creator

Data in Triple Store

Figure 5 Process flow from Triple Store to Visual Analytics

The purpose of the Context Creator is to convert data into a formal context by FCA “scaling”. The user uses the Context Creator to query the triple store to select the data to convert. The user can also decide how to scale the data. The Context Creator has the following functionality:

* Handling a wide range of data types for attributes, including categorical, Boolean, continuous, ordinal and date/time.
* Support of Discrete and Progressive FCA scaling for continuous, ordinal and date attributes.
* Display of meta-data (attributes and attribute values, and data types) in GUI.
* User-creation of queries based on selection of attributes and attribute values in the GUI.



## Visual Analytics

In the beginning of the project, based on the nature of each use case analysis, CRSA has implemented several alternatives for the visual exploration of concept lattices through searching, filtering sub-selection of concepts and attributes; visual display of related attributes and their implication confidences. A web-based visual analytics prototype was developed to implement all the functions mentioned, called CUBIX. Typical uses of CUBIX include semantic data analysis and pattern detection, anomaly detection, comparisons, information classification, and knowledge discovery. CUBIX’s workflow allows users to carry out an analysis starting from a real data set, converting it into a formal context, simplifying the context to make it manageable, and visualizing the result as a concept lattice.

To illustrate the features of the first version of CUBIX to the use-case partners, CRSA has used a sample of a traffic accident dataset occurred in the UK in 2006 and conducted an analysis for most common causes between different combinations of attributes (e.g. road surface, weather, accident severity, etc). FCA allowed us to answer pattern identification questions such as “What can be considered the main causes of ‘accident severity: serious’?” or “how many accidents are ‘light conditions: darkness’ and ‘accident severity: serious’?".

In the next phase, a number of new analytic features in the new version of CUBIX have been integratedCUBIST analytics feature (Figure 9). We briefly describe each of them below.

*Charts*: In addition to the main concept lattice visualization, several charts display different aspects of the underlying conceptual structure such as co-occurrence of attributes, concepts distribution, stability vs. support, etc. (Figure 1 - 3). Some charts are updated when the user points the mouse over a concept, highlighting details of the concept. Similarly, a selection of a point/series in the chart will highlight the concerned concepts in the lattice. This technique is called *Linking and Brushing.*

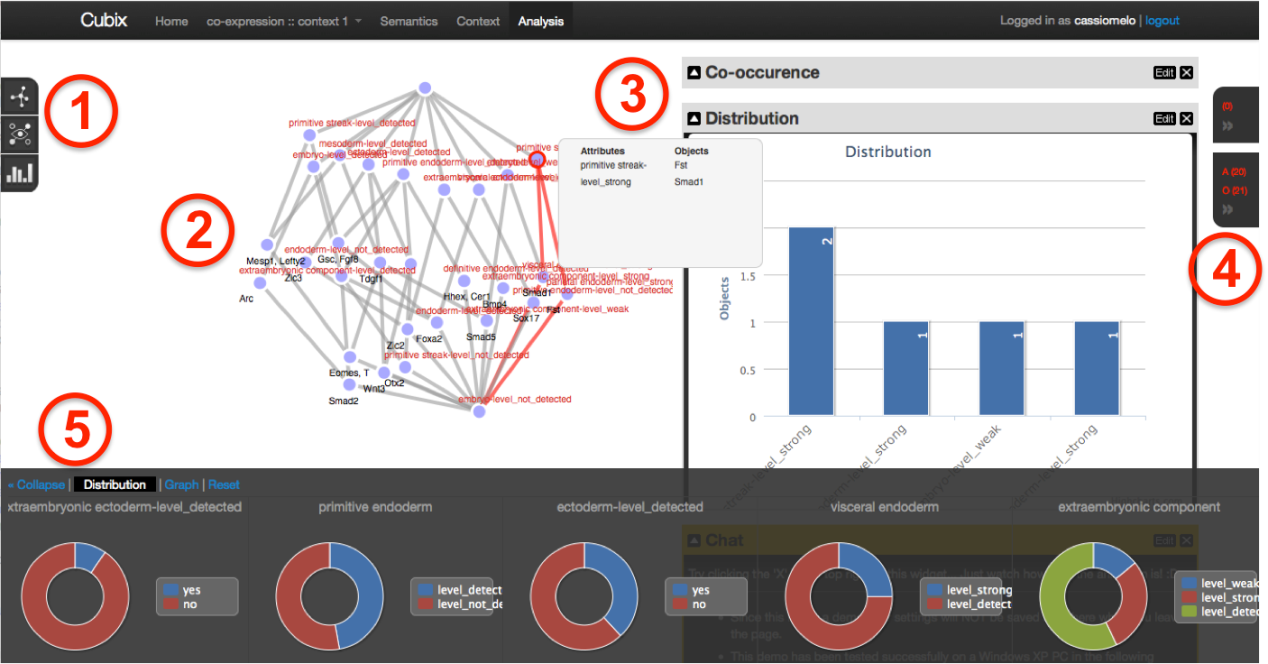


Fig. 9. Cubix user interface displaying the adult data set. Its main components: 1) Toolbar; 2) Visualization canvas; 3) Dashboard; 4) Selection & entities bar and; 5) Filter bar.

*Association Rules Analytics*. Association Rules are of the form premise ⇒ conclusion and carry very little information about how they can be visualized. They are typically displayed as a list of logical sentences, unpractical to analyze when the number of rules is large. CUBIST now provides three visualizations for association rules, combined with statistics and charts to enable progressive exploration of the rules set: A matrix view, where each rule is displayed in a row and the concerned pairs of attribute-value in columns, with purple cells representing the premise and yellow ones the conclusion (Figure 2). The second visualization is a radial graph showing how pairs attribute-value implies to each other in a radial graph layout. Finally, a bubble graph visualization displays premises and conclusions as connected bubbles with the concerned attribute-value pairs inside each bubble. A scatterplot matrix in the dashboard shows the distribution of confidence, support and lift for the association rules, allowing users to graphically select portions of the distribution they are interested in.

*Transformations*. This option allows users to transform the concept lattice in other structures like a tree or bi-partite graphs that are easier to navigate than lattices. Trees are common and have easily understandable visual representations. We consider them as a visualization alternative to large cluttered concept lattices, which preserves all lattice entities and some of its structure.

*Concept Clustering*. Clustering of concepts can be useful to facilitate the analysis and to identify zones of interest. Cubix uses a spectral/k-means clustering algorithm to identify clusters. Some similarity measures are based on the concept lattice topology (e.g. counting the number of links between two concepts); Intent/extent similarity (e.g. Jaccard); or confidence between two pairs of concepts.

In the last phase of the project, particularly based on data collected from user evaluations (detailed in D.1.4.2), emphasis was laid on many usability improvements were made based on the use cases feedback, e.g. zooming in the visualization, tooltips to explain specific features, etc. Some visualisations proposed in previous versions were removed for not being effective to our Use Cases data. Perhaps the most noticeable change concerns the labelling of nodes in the visualisations. Those were one of the main complaints from users, since formal concepts in particular are identified in terms of both its attributes and objects and thus it may lead to cumbersome visualisations. On the other hand, each visualisation requires a different labelling positioning strategy. In the example of Figure 10 below, the concept lattice drawing algorithm now takes into account the width and height of the label box in order to position nodes properly. Similar techniques were employed for other visualisations like the sunburst.

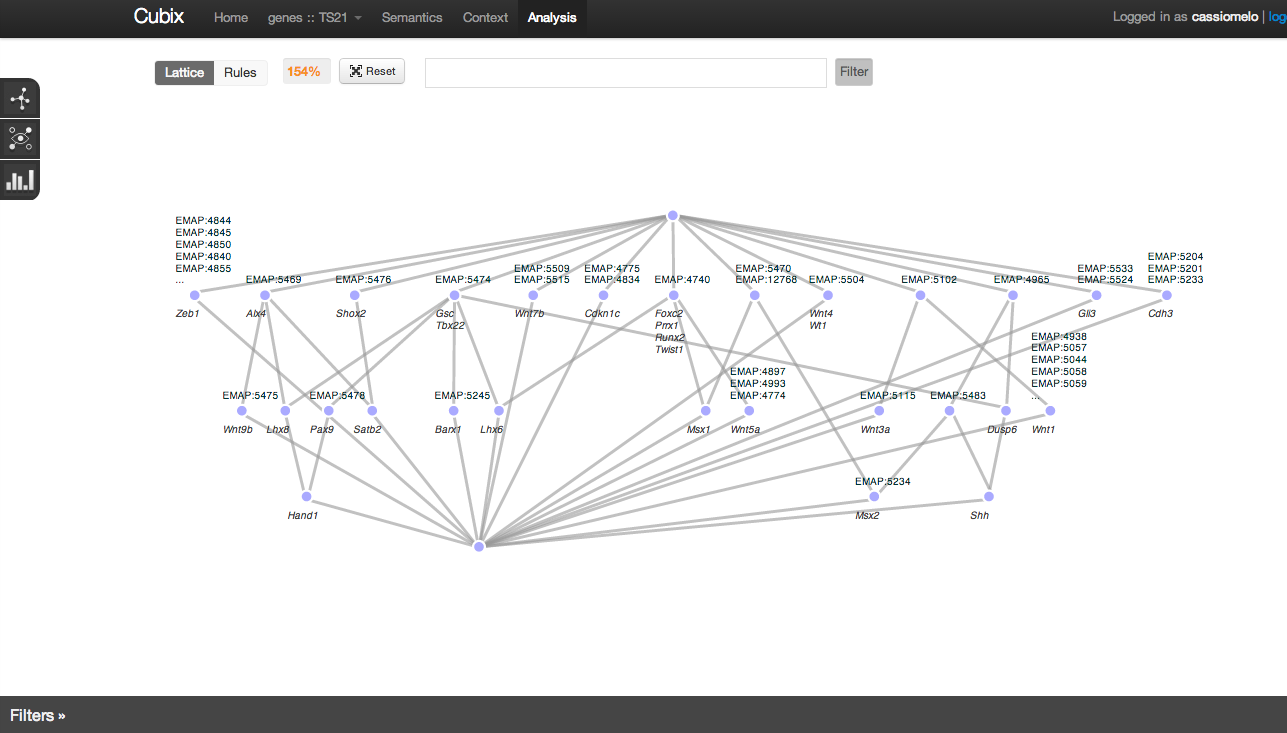
**

Fig. 10. Improved labels positioning for Concept Lattice Visualisation.

# Outcome

## User Evaluation

* + - CUBIST is an expert tool
    - Having and Integrating (!) different infovis means is good! Main work hypothesis of CUBIST supported. THIS is beyond SoA
    - BUT: Issues with ease of use
    - Hasse pays off
    - Graph-Exploration view is good

In the last period of the project, an overall CUBIST evaluation had been conducted. For each use case, two test users had to walk through use case specific tasks. After this, structured interviews with the test users have been conducted, and they had to fill a questionnaire. The main results of the evaluation are given in the remainder of this section.

First of all, we distinguished in the evaluation between two kinds of use case partners. One use case partner, namely Innovantage (INN), has more traditional information needs, i.e. needs which are met by traditional BI-tools. The other two use case partners, Heriot-Watt-University (HWU) and Space Applications Services (SAS), want to analyse the data in a different understanding. They are less interested in some KPIs (key performance indicators), measured by numbers, of their data. Instead, they want to investigate patterns or clusters in their data, or they want to scrutinize dependencies between different attributes.

As it is in the nature of a research project, CUBIST investigated how to meet more novel information needs. From a birds eye perspective, CUBIST is less positively rated by the INN users than by the HWU/SAS users. In other words: CUBIST is better suited for novel ways to analyse information and less suited for the traditional “show me the numbers”-approach. This supports the CUBIST hypothesis that the CUBIST approach is not improving, but complementing traditional BI-means.

A main feature of CUBIST is the integration of different means which allow to search for data (factual search), explore the data (explorative search) and analyse the data (visual analytics). The evaluation shows that supporting these different types of information need is appreciated by the users, and each of the corresponding components is rated useful for specific information needs. The evaluation shows furthermore that in CUBIST, from a usability point of view, integrating different components and visualizations is challenging.

Second, the visual analytics means in CUBIST are dominated by novel visualisations which show clusters and dependencies instead of numbers, linked to some traditional visualisations. The visual analytic features target the more novel information needs. Having different types of visualisations integrated in one BI-tool is positively rated, even if some visualisations are in the beginning hard to understand. The evaluation showed clear downsides of the “Visual Analytics” component as well. Its ease of use, appeal and attractiveness are badly rated.

*Finally, the user’s consider CUBIST an expert tool. The novel approach taken in CUBIST to select, transform and visually display data, particularly some novel visualisations, need some learning effort in the beginning. The evaluation shows that those experts who have taken that effort appreciate that this finally paid off.*

## Our Take

m a user’s point of view, the following two features are essential:

1. CUBIST provides components which support factual search (“Search & Select”), explorative search (“Navigate in Data”, “Explore Selection) and visual analytics features (“Analyse Selection”).
2. The component of CUBIST in turn features different types of Visual Analytics means, like traditional BI visualizations (bar charts), novel tree-based visualizations (e.g. sunburst diagrams), FCA- and graph-based diagrams (e.g. the Hasse Diagrams), which are interlinked and highly interactive.

For this reason, the evaluation focused not only on evaluating the overall prototype, but it evaluated the different components separately (with an emphasis on the “Analyse Selection” component) and their integration in the prototype.

During the evaluation, the users had to walk through a set of use-case-specific tasks. Then we used interviews for gathering qualitative information and questionnaires for gathering quantitative information from the users. For each part of the evaluation (evaluation of the overall prototype, evaluation of the integration of the different components, evaluation per component) we provided in the previous section a short summary and conclusion.

Ii this final chapter, an overall evaluation summary is given. Based on the conclusions of the evaluation, we propose possible improvements of future BI tools.

First of all it has to be noted that CUBIST is perceived an expert tool by the users. It targets experts for a given domain of discourse who

1. Have specific information needs and who need some sort of in-depth analysis of their data
2. who have enough skills and time resources to familiarize with the novel and sophisticated Visual Analytics means in CUBIST.

Thus the recommendations provided in this section should be understood to target BI tools for experts, not tools for casual BI users. Moreover, the recommendation target future BI-tools which aim at addressing information needs which go beyond those needs which can already be addressed by traditional tools.

First of all, the evaluation shows that different tasks (with different information needs) in different use cases and require different means in a tool to meet them. Apart from the “Navigate in Data” component, each component is considered useful for some information needs. The evaluation shows that the integration of components which support factual search, explorative search and Visual Analytics pays off. The evaluation shows furthermore that in CUBIST, from a usability point of view, integrating different components and visualizations is challenging.

**Proposed recommendation 1:** Future BI tools should not only focus on the analysis (in the BI understanding) of data, but on the search in data and the exploration of data as well. Integrating different components which target different information needs is challenging and needs further investigations.

The “Search and Select” component uses a faceted search approach to search for and filter the data. Faceted search frontend are meanwhile quite common, and unsurprisingly, the users do not consider this component very novel, but easy to use. Moreover, this component is considered the most useful one in CUBIST, and it is understood to be the entry point for all further activities in CUBIST. Note finally that the filtering-facilities of this component are appraised by the users, though users had concrete recommendation on how this component can be improved.

**Proposed recommendation 2:** It is very reasonable to have faceted search based frontend in future BI-solutions for searching and filtering the data. The evaluation gives clear hints on which filtering functionalities are requested by the users.

The “Explore Selection” component is considered innovative and useful, and it has a clear purpose. The test users disagree in its ease of use. The conclusion of serves as our next recommendation:

**Proposed recommendation 3:** Future BI solutions, which aim at providing means to explore the data, should incorporate functionalities which resemble the functionalities of the “Explore Selection” Component. Designing the interface for such exploration means deserves closer attention.

Finally, we consider the Visual Analytics features of CUBIST. Next, we summarize those features and provide corresponding recommendations.

CUBIST features a variety of novel visualisations (sunburst, sankey, matrix…), combined with relatively simple, traditional visualizations (bar chart, pie charts). The novel visualisation features are a clearly dominant feature of the “Analyse Selection” component, and they target different information needs as satisfied by traditional BI tools. They complement traditional BI: CUBIST is strong when it comes to finding clusters and relations in the data, but is not well suited for traditional BI use cases. Having different types of visualisations integrated in one BI-tool is positively rated, even if some visualisations are in the beginning hard to understand. Surprisingly, particularly Hasse-diagrams are appraised by the test users.

The evaluation showed clear downsides of the “Visual Analytics” component as well. Its ease of use, appeal and attractiveness are badly rated. Some reasons for these bad ratings can be found in the interviews: The component is too crowded with visualizations or options, and the way how different visualisations interact is in CUBIST not well solved.

This leads after all to the final recommendation:

**Proposed recommendation 4:** Future BI-tools should comprise quite different Visual Analytics means, ranging from traditional to novel ones (e.g. graph-based). One should not hesitate to include unfamiliar, sophisticated visualizations into expert BI tools, even if those visualizations are not ease to digest from the very beginning.

Care has to be taken on how in such future tools different components and visualizations are integrated and how they interact, and further research on how one can integrate different information access means in one tool in a user-friendly, understandable way.

## Conclusions

Lorem ipsum blablaa

# Bibliography

From D1.2.1 (architecture)

(Choo1998), Choo, C. W.; Detlor, B. & Turnbull, D. *A Behavioral Model of Information Seeking on the Web -- Preliminary Results of a Study of How Managers and IT Specialists Use the Web*, ASIS Annual Meeting Contributed Paper, 1998

(DODAF2010) DOD Architecture Framework, v.2.02, <http://cio-nii.defense.gov/sites/dodaf20>, 2010

(Ellis1989) Ellis, D. *A behavioural model for information retrieval system design*, J. Inf. Sci., Sage Publications, Inc., 1989, 15, 237-248

(Ellis1989a) Ellis, D. *A behavioral approach to information retrieval system design*, J. Doc., Aslib, The Association for Information Management, 1989, 45, 171-212

(Mahmood2011) Mahmood, Z.; Hill, R. *Cloud Computing for Enterprise Architectures*, Springer, 2011, pp.12

(TOGAF2009) The Open Group: TOGAF Version 9 Enterprise Edition, <https://www2.opengroup.org/ogsys/jsp/publications/PublicationDetails.jsp?catalogno=g091>, 2009.

(Zachman1999) Zachman, J.: *A framework for information systems architecture*. IBM Syst. J. 38 (2/3), 454–470, 1999

(InClose2) In-Close2, a High Performance Formal Concept Miner. In: Andrews, S., Polovina, S., Hill, R. and Akhgar, B. (eds.): Conceptual Structures for Discovering Knowledge - Proceedings of the 19th International Conference on Conceptual Structures (ICCS) 2011. LNAI 6828. pp. 50-62. Springer - Berlin, 2011.

(FcaBedrock) Andrews, S and Orphanides, C.: Knowledge Discovery through creating Formal Contexts. In: Xhafa, F. (ed.): Int. J. of Space-Based and Situated Computing (IJSSC), Inderscience, (accepted article).

(Cubix) Melo, C.A., Aufaure, M.-A., Bezerianos, A. and Le Grand, B.: Cubix: A Visual Analytics Tool for Formal Concept Analysis. In: 23ième Conférence Francophone Sur l’IHM (IHM 2011) – Demo. Sophia-Antipolis, France, 2011.

**Use Cases**

1. <http://www.talend.com/products/open-studio-di.php> [↑](#footnote-ref-1)
2. <http://d2rq.org/d2r-server> [↑](#footnote-ref-2)