Deliverable 4.2: Initial responsive platform prototype, modules and common communication protocol

Sebastian Gottfried/TUD
Alexandros Pournaras, Chrysa Collyda, Vasileios Mezaris/CERTH
Tobias Backes/GESIS
Alfred Wertner/KNOW
Aitor Apaolaza/UMAN
Till Blume, Ahmed Saleh/ZBW

29/09/2017

Work Package 4: Iterative MOVING platform development

TraininG towards a society of data-saVvy inforMation prOfessionals to enable open leadership INnovation

Horizon 2020 - INSO-4-2015

Research and Innovation Programme

Grant Agreement Number 693092

Dissemination level	PU
Contractual date of delivery	30/09/2017
Actual date of delivery	29/09/2017
Deliverable number	4.2
Deliverable name	Initial responsive platform prototype, modules and common communication protocol
File	`moving_d4.1-v1.0.pdf`
Nature	Report
Status & version	Draft v1.0
Number of pages	43
WP contributing to the deliverable	4
Task responsible	TUD
Other contributors	CERTH, GESIS, KNOW, UMAN, ZBW
Author(s)	Sebastian Gottfried/TUD Alexandros Pournaras, Chrysa Collyda, Vasileios Mezaris/CERTH Tobias Backes/GESIS Alfred Wertner/KNOW Aitor Apaolaza/UMAN Till Blume, Ahmed Saleh/ZBW
Quality Assessors	ZBW
EC Project Officer	Hinano SPREAFICO
Keywords	Prototype, Responsive Design, MOVING platform

Table of contents

List of Figures

List of Tables

Executive summary

The aim of of this document is to report the development results of the MOVING platform up to month 18 of the project. The MOVING platform provides an integrated working and training environment for data professionals. This deliverable presents how the architecture designed in Deliverable D4.1 “Definition of platform architecture and software development configuration” has been realized. Furthermore, we detail the status of the individual components of the MOVING platform and show their interim results.

Abbreviations

Abbreviation	Explanation
API	Application Programming Interface
ATS	Adaptive Training Support
cMOOC	*connectivist* Massive Open Online Course
CSV	Comma-Separated Values
DCMI	Dublin Core Metadata Initiative
DOM	Document Object Model
FDC	Focused web Domain Crawler
GUI	Graphical User Interface
GVF	Graph Visualisation Framework
HTML	HyperText Markup Language
JS	JavaScript
JSON	JavaScript Object Notation
LOD	Linked Open Data
MOOC	Massive Open Online Course
QA	Quality assurance
RDF	Resource Description Framework
REST	REpresentational State Transfer
SEC	Search Engine-based web Crawler
SPARQL	SPARQL Protocol and RDF Query Language
SSM	Social Stream Manager
TOC	Table Of Contents
TS	TypeScript
URL	Uniform Resource Locator
VPN	Virtual Private Network
XML	Extensible Markup Language
WebGL	Web Graphics Library
WevQuery	Web Event Query Tool

1 Introduction

1.1 History of the document

Date	Version
17.08.2017	v0.1: Initial TOC draft
21.08.2017	v0.2: Second TOC draft
08.09.2017	v0.3: QA ready
22.09.2017	v0.9: QA comments addressed
28.09.2017	v1.0: Final version

1.2 Purpose of the document

This document describes the progress in the implementation of the initial MOVING platform prototype. First, we give an overview of the platform architecture (Section 2). Then we describe the components which have been integrated for MOVING platform. This includes the MOVING web application (Section 3), the search engine (Section 4), the user training support (Section 5), the user behaviour analysis (Section 6) and the data acquisition and processing components (Section 7).

2 Platform overview

In this section, we introduce the architecture of the MOVING platform. The platform is currently installed on a server at the TU Dresden. Users can access the platform under the address https://moving.mz.test.tu-dresden.de/.[1] The platform architecture has been developed in accordance to the requirements collected in Deliverable D1.1 "User requirements and Specification of the use cases". Based on this, other possibilities that were briefly considered before the start of the project, such as a SCORM module, were found to be out of scope of the MOVING project.

Figure 1 shows the most important components of the platform and their relationships.

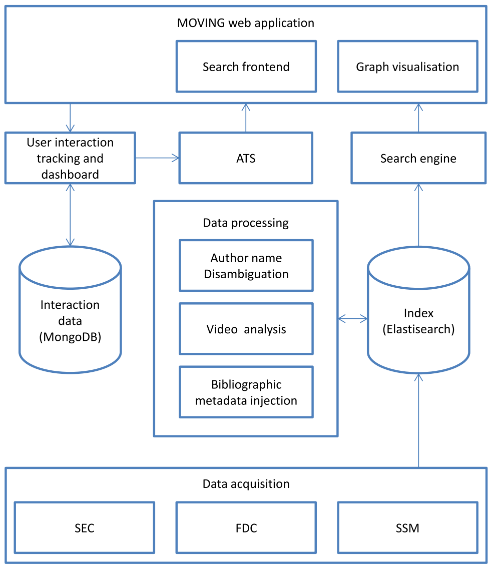


Figure 1: Overview of the interactions between the different components of the MOVING platform

The core of the platform is the MOVING web application as described in Section 3. It holds the user interface of the application. Part of the web application are the search frontend and the Graph Visualisation Framework which is used to provide advanced presentations of search results as illustrated in Section 3.1 and

[1]Currently the access is limited to network of the TU Dresden. We can also provide access to the platform for interested parties outside of the TUD network via a VPN connection. Please contact Sebastian Gottfried (sebastian.gottfried@tu-dresden.de) to get access to the platform. We are working on making the platform available to the general public.

Section 3.2. The web application pulls data from two main data sources: the MOVING search engine (see Section 4) and the Adaptive Training Support (ATS) service (see Section 5). The former is responsible for searching for documents, while the latter addresses the users' training.

The ATS service depends on detected user behaviour patterns. The user interaction tracking and dashboard (Section 6) collects the necessary interaction data and stores it in a database. The dashboard allows us to define complex interaction patterns to be detected in the stream of collected low-level events. The detected patterns are fed into the ATS service.

The remaining components of the MOVING platform are used to acquire and process data for the search index (Section 7). There are three crawlers to collect new data:

– the Focused web Domain Crawler (FDC), which crawls specific web pages and is introduced in Section 7.1;

– the Search Engine-based web Crawler (SEC), which discovers new web pages and is presented in Section 7.2;

– the Social Stream Manager (SSM) to crawl social media, as shown in Section 7.3.

Additionally, we have three services to enhance the data stored in the search index:

– the bibliographic metadata injection (Section 7.4);

– the author name disambiguation (Section 7.5);

– the video analysis service (Section 7.6).

With these components realized we have implemented the MOVING platform technological novelties, in accordance with the analysis of the platform's envisaged innovation potential and contribution beyond state of the art, which was presented in Section 4 of Deliverable D2.4 "Open innovation systems state of the art and beyond".

3 MOVING web application

The MOVING web application is an extended version of pre-existing *eScience Platform*[2] of the TU Dresden. We have extended the application with an frontend for the MOVING search engine (see Section 3.1), novel graph visualisations (see Section 3.2) and a new responsive design (see Section 3.3).

3.1 Search frontend

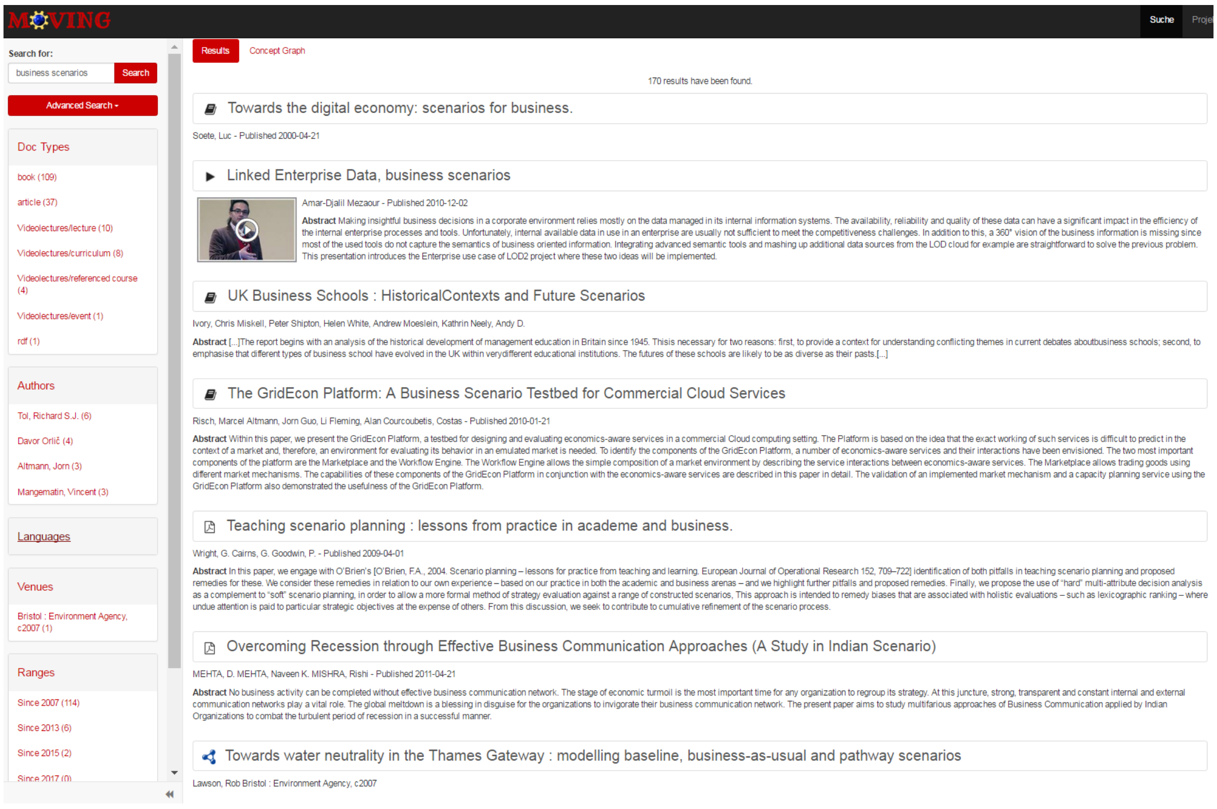
In order to help the users of the MOVING platform to find the most relevant results to their search query, we developed a user-friendly frontend which consists of four main components: search bar, advanced search form, faceted search widgets and search results form. In order to design a user-friendly interface, our partner Ernst & Young (EY) provided a detailed user experience style guide. The main purpose of the style guide is to create a foundation for all web applications and digital tools which keep a uniform user experience across all interfaces. This is an important part of usability and will help the potential users of the MOVING platform, especially from EY, to quickly adapt and use our search frontend. The user experience style guide consists of eleven chapters. In the following, we highlight how we reflected the main style guide instructions in our platform:

1. Iconography: we used a set of icons to demonstrate the search result types as illustrated in Table 1.

📖	📄	●	▶	@
Book	Paper	RDF	Video	Website

Table 1: Icons for different documents types

[2]https://www.escience-sachsen.de/, last accessed at 21.09.2017



Search for: business scenarios Search

170 results have been found.

Towards the digital economy: scenarios for business.

Soete, Luc - Published 2000-04-21

Linked Enterprise Data, business scenarios

Amar-Djalil Mezaour - Published 2010-12-02

Abstract Making insightful business decisions in a corporate environment relies mostly on the data managed in its internal information systems. The availability, reliability and quality of these data can have a significant impact in the efficiency of the internal enterprise processes and tools. Unfortunately, internal available data in use in an enterprise are usually not sufficient to meet the competitiveness challenges. In addition to this, a 360° vision of the business information is missing since most of the used tools do not capture the semantics of business oriented information. Integrating advanced semantic tools and mashing up additional data sources from the LOD cloud for example are straightforward to solve the previous problem. This presentation introduces the Enterprise use case of LOD2 project where these two ideas will be implemented.

UK Business Schools : HistoricalContexts and Future Scenarios

Ivory, Chris Miskell, Peter Shipton, Helen White, Andrew Moeslein, Kathrin Neely, Andy D.

Abstract [...]The report begins with an analysis of the historical development of management education in Britain since 1945. Thisis necessary for two reasons: first, to provide a context for understanding conflicting themes in current debates aboutbusiness schools; second, to emphasise that different types of business school have evolved in the UK within verydifferent educational institutions. The futures of these schools are likely to be as diverse as their pasts.[...]

The GridEcon Platform: A Business Scenario Testbed for Commercial Cloud Services

Risch, Marcel Altmann, Jorn Guo, Li Fleming, Alan Courcoubetis, Costas - Published 2010-01-21

Abstract Within this paper, we present the GridEcon Platform, a testbed for designing and evaluating economics-aware services in a commercial Cloud computing setting. The Platform is based on the idea that the exact working of such services is difficult to predict in the context of a market and, therefore, an environment for evaluating its behavior in an emulated market is needed. To identify the components of the GridEcon Platform, a number of economics-aware services and their interactions have been envisioned. The two most important components of the platform are the Marketplace and the Workflow Engine. The Workflow Engine allows the simple composition of a market environment by describing the service interactions between economics-aware services. The Marketplace allows trading goods using different market mechanisms. The capabilities of these components of the GridEcon Platform in conjunction with the economics-aware services are described in this paper in detail. The validation of an implemented market mechanism and a capacity planning service using the GridEcon Platform also demonstrated the usefulness of the GridEcon Platform.

Teaching scenario planning : lessons from practice in academe and business.

Wright, G. Cairns, G. Goodwin, P. - Published 2009-04-01

Abstract In this paper, we engage with O'Brien's [O'Brien, F.A., 2004. Scenario planning – lessons for practice from teaching and learning. European Journal of Operational Research 152, 709–722] identification of both pitfalls in teaching scenario planning and proposed remedies for these. We consider these remedies in relation to our own experience – based on our practice in both the academic and business arenas – and we highlight further pitfalls and proposed remedies. Finally, we propose the use of "hard" multi-attribute decision analysis as a complement to "soft" scenario planning, in order to allow a more formal method of strategy evaluation against a range of constructed scenarios, This approach is intended to remedy biases that are associated with holistic evaluations – such as lexicographic ranking – where undue attention is paid to particular strategic objectives at the expense of others. From this discussion, we seek to contribute to cumulative refinement of the scenario process.

Overcoming Recession through Effective Business Communication Approaches (A Study in Indian Scenario)

MEHTA, D. MEHTA, Naveen K. MISHRA, Rishi - Published 2011-04-21

Abstract No business activity can be completed without effective business communication network. The stage of economic turmoil is the most important time for any organization to regroup its strategy. At this juncture, strong, transparent and constant internal and external communication networks play a vital role. The global meltdown is a blessing in disguise for the organizations to invigorate their business communication network. The present paper aims to study multifarious approaches of Business Communication applied by Indian Organizations to combat the turbulent period of recession in a successful manner.

Towards water neutrality in the Thames Gateway : modelling baseline, business-as-usual and pathway scenarios

Lawson, Rob Bristol : Environment Agency, c2007

Figure 2: Screenshot of the search results page

2. Responsive Design: It enables the platform to deliver a uniform experience to multiple devices and screen sizes. More details are provided in Section 3.3.
3. Forms: The text fields and labels of the forms has been laid out according to the guide to ensure large enough click or touch targets on all support devices.

Video.Lecture	Video.Debate	Video.Demonstration	Video.discussion
Video.Interview	Video.Introduction	Video.Course	Video.Opening
Video.Invitation	Video.Announcement	Video.Keynote	Video.Self Introdution
Summary	Tutorial	Press Conference	Video Conference
Video.Advertisment	Video.Invited Talk	Video.Panel	Video.Poster
Video.Best Paper	Video.Demonstration	Video.Promotional Video	Video.Thesis Proposal
Video.Thesis Defense	Video.External Lecture	Video.Event	Video.Event Section
Video.Event(ToC)	Video.Event - as Course	Video.Project	Video.Project Group
Video.Session	Video.Referenced Course	Video.Curriculum	Video.Default
RDF	Book	Website	Article

Table 2: Document types in the MOVING platform v1.0

As previously mentioned, the MOVING platform consists of different datasets with different document types. Another important stride towards allowing the users to filter the search results efficiently, is the faceted search functionality. Figure 2 shows a screenshot of the faceted search widgets in the left side bar. The user can easily filter the results based on the document type, authors, languages, venues, and/or dates ranges. Table 2 shows the various types of documents which have been integrated in the MOVING platform.

3.2 Graph visualisation

Integrating a graph visualisation into the MOVING platform should support the user in analysing the retrieved search results as well as identifying further valuable documents by navigating and filtering the visualised graph. Whenever the user searches for documents (see Section 3.1), a list of results is presented. Each result might have different attributes like authors, languages, document types, etc. This allows to build up a network between the documents and their attributes and to visualise them using an interactive, web-based graph visualisation. This graph visualisation is built on top of the *Graph Visualisation Framework* (GVF), developed at the Know-Center and used in different projects. The following sections describe the functionality and structure of GVF and how it is integrated into the MOVING platform.

3.2.1 Overview

Using the faceted search of the MOVING platform provides a paginated list of results. On the top of the result list, the user can switch to the graph visualisation using the `Concept Graph` button. This leads to a replacement of the result list by the integrated graph visualisation as Figure 3 shows.

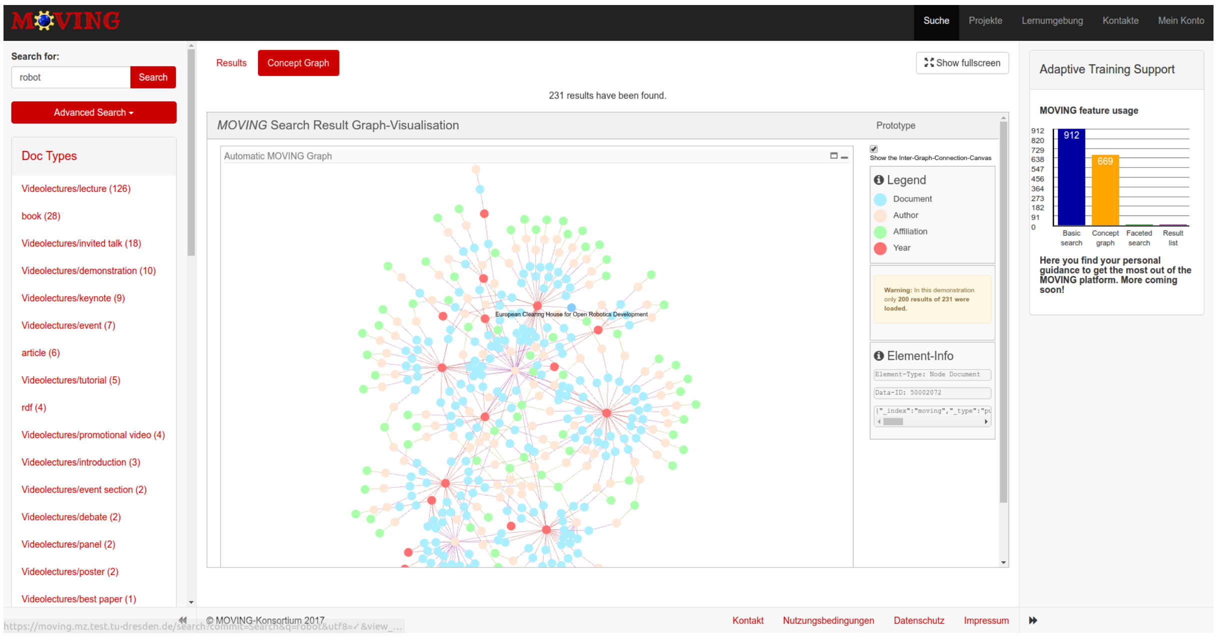

MOVING
Suche
Projekte
Lernumgebung
Kontakte
Mein Konto
Search for:
robot
Search
Advanced Search ▾
Doc Types
Videolectures/lecture (126)
book (28)
Videolectures/invited talk (18)
Videolectures/demonstration (10)
Videolectures/keynote (9)
Videolectures/event (7)
article (6)
Videolectures/tutorial (5)
rdf (4)
Videolectures/promotional video (4)
Videolectures/introduction (3)
Videolectures/event section (2)
Videolectures/debate (2)
Videolectures/panel (2)
Videolectures/poster (2)
Videolectures/best paper (1)
Results
Concept Graph
Show fullscreen
231 results have been found.
MOVING Search Result Graph-Visualisation
Prototype
Automatic MOVING Graph
European Cleaning House for Open Robotics Development
Show the Inter-Graph-Connection-Canvas
Legend
Document
Author
Affiliation
Year
Warning: In this demonstration only 200 results of 231 were loaded.
Element-Info
Element-Type: Node Document
Data-ID: 50002072
{"_index":"moving","_type":"pu
Adaptive Training Support
MOVING feature usage
912
820
729
638
547
456
364
273
182
91
0
912
669
Basic search
Concept graph
Faceted search
Result list
Here you find your personal guidance to get the most out of the MOVING platform. More coming soon!
© MOVING-Konsortium 2017
Kontakt
Nutzungsbedingungen
Datenschutz
Impressum

Figure 3: Search results visualised as a graph of documents, authors, affiliations and corresponding years.

3.2.2 Graph visualisation framework

The Graph Visualisation Framework (GVF) was developed during the last year with the purpose to provide a framework which can be used in multiple projects to visualize interactive graphs as a web-application. The development was driven by the following motivations:

Performance Several open-source frameworks exist which easily can be used to visualise graphs in a web application. However, most of them are SVG[3]-based or are rendering directly on a HTML5 canvas[4]. These technologies have a limited performance, since they are not suitable to render lot of nodes and edges simultaneously. Using WebGL[5] for rendering elements makes use of the graphics card acceleration. Thus, we decided to build up a framework from the scratch. We use the THREE.js[6] library as an abstraction over WebGL which gives us uncomplicated method to access the features of WebGL.

Reusability It was clear from the beginning that the framework is going to be used in multiple projects. That's why we decided to split the implementation into a base library (GVF-Core) and project related code. The core is usable as a library, and is extendible and derivable. The latter requirements are challenging, since frontend web applications are commonly written in JavaScript. Deriving classes and object oriented programming in general is not easy when using classical JavaScript. Thus, we decided

[3]https://www.w3.org/TR/SVG11/, last accessed at 22.09.2017
[4]https://html.spec.whatwg.org/multipage/canvas.html#the-canvas-element, last accessed at 22.09.2017
[5]https://www.khronos.org/registry/webgl/specs/latest/, last accessed at 22.09.2017
[6]https://threejs.org/, last accessed at 22.09.2017

to write GVF in TypeScript[7], a programming language which is based on the ECMAScript 6[8] definition. It supports classes and inheritance, and can be transpiled to pure JavaScript.

Modularity GVF needs to be built as a modular framework, where existing libraries are easy to include and extending is as easy as possible. Thus, we decided to rely on the Angular2[9] framework.

Figure 4 gives an overview of how GVF is structured. It shows that GVF is not a single implementation but consists of two big parts: A) The GVF-Core and B) the implementation for MOVING.

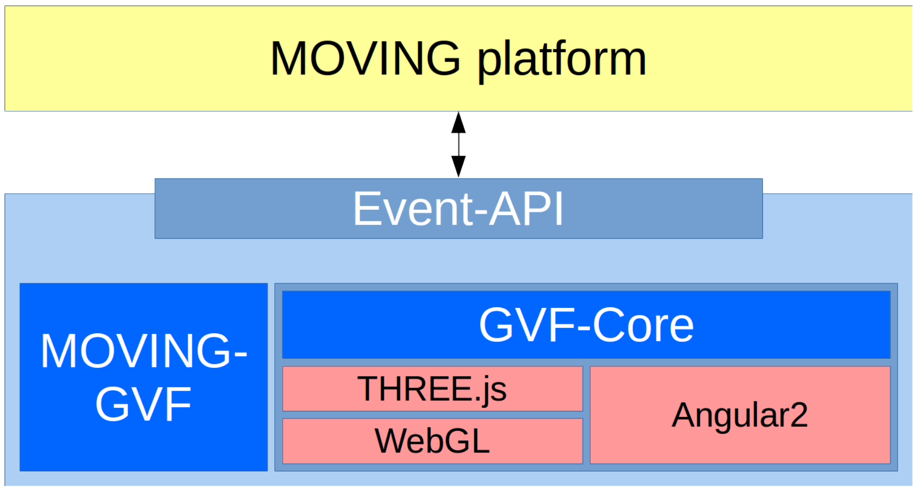


Figure 4: Overview of the GVF environment and how it communicates the MOVING platform

The GVF core is organised as a separate project[10]. It can not be executed as a standalone application but can be used as a extendible library. Some of the classes in the core are directly usable in an application, others are implemented as abstract classes and have to be defined in a derived, domain specific class. The MOVING specific part[11] of GVF holds the core as a GIT submodule. The rest of the code contains data- and communication specific parts, derivations of different abstract data types to e.g. define MOVING specific entities and nodes.

The following list provides a basic overview of the most important classes in the core:

Entities & Connections The GVF strictly separates data and graphical representation. The main reason is, that a data entity (e.g. a document) might be visualised as a node on multiple planes (see below) and thus, different graph-elements on multiple planes can be linked to the same data entity. Similar to the entities are the *connection*. They define the logical relation between two entities and might be represented as edges in a graph.

Planes & Graphs GVF allows to show multiple, interactive graphs simultaneously. Those are organized on *planes* which each further contain an instance of the *Graph* class. This class holds and organises a specific layout, the underlying THREE.js renderer and scene, as well as all nodes, edges and further elements. It is further responsible to generate nodes and edges from the data entities and link them. This could be done manually or by deriving the `AutoGraph` class, which just has to know which entity should be visualised as which node-type and which connection should generate an edge. Planes are responsible to show the graph on the right position and size inside the application.

Layouts Graphs can be visualised using different layouts. For example a layout class using a force-directed placement algorithm can be used, which places the nodes in a way that the edges between the nodes have a similar length and avoids crossing edges as much as possible.

Graph elements All nodes, edges, meta nodes and paths are derived from the `GraphElementAbstract` class, which is used as a THREE.js container to hold the geometry primitives. It is further responsible for catching events like hovering or clicking on it.

[7]https://www.typescriptlang.org/, last accessed at 21.09.2017
[8]https://www.ecma-international.org/ecma-262/6.0/index.html, last accessed at 22.09.2017
[9]https://angular.io/, last accessed at 22.09.2017
[10]https://git.know-center.tugraz.at/summary/?r= phasitschka/gvf/core.git, last accessed at 22.09.2017
[11]https://git.know-center.tugraz.at/summary/?r= phasitschka/gvf/moving.git, last accessed at 22.09.2017

Nodes & Edges As described above, entities and connections can be visualised using instances of node and edge classes. Each node is derived from `NodeAbstract` and each edge from `EdgeAbstract`.

Meta nodes GVF also provides meta nodes which summarize or aggregate a set of nodes. For example, MOVING uses the `StarChart` for aggregating nodes and their attributes and the `OnionVis` as a base for the navigation concept.

Labels Applying text on a WebGL scene is not as trivial as in a simple HTML based application. Thus, we implemented a class which allows to place HTML-Text elements above the THREE.js scene graph. It takes the global scene coordinates and camera positions into account.

Services The interactivity, animations and communication between different graphs are solved by so called services in Angular2. They can be accessed from every component globally.

Plugin API The plugin system is responsible to provide an entry-point for the domain-specific application to load data and create planes and graphs.

The MOVING-specific code is structured as follows:

MovingPluginApi This is the entry point for the moving specific application part. It registers an event, which fires when the search result data is available (see Section 3.2.3). The data is then stored as entities and connections, and a plane containing an instance of `MovingAutoGraph` is created to start the visualisation.

MovingDataSourceMovingPlatform This class is responsible for converting raw data from the MOVING platform to GVF entities like `DocumentDataEntity` or `AuthorDataEntity`. It further creates connections between the entities (e.g. `DocAuthorConnection`).

MovingAutoGraph This class extends the `AutoGraph` class from the core to convert the entities to the specific nodes and edges which are visible in the visualisation.

Event API This component implements event-based communication between the GVF and the MOVING platform. Section 3.2.3 provides a detailed description.

gvf.html This file initializes the visualisation by loading the application JavaScript code.

3.2.3 Deployment and integration of the framework into the MOVING platform

GVF is written in TypeScript using Angular2 framework. Thus, the source code cannot be just run in the browser but needs a preprocessing step. This includes the transpiling from TypeScript to JavaScript. The output of this process gets included into the `/public/gvf` folder of the MOVING platform, which makes the files accessible via an URL. The graph view of the search frontend `gvf.html` as an iframe. Additionally the Event-API JavaScript file (`gvfapi.js`) is loaded to make a communication with GVF possible.

GVF runs in an iframe, since this allows it to be included in any third party environment without the risk of conflicts of libraries or frameworks. However, the variables, objects or classes of the iframe content (GVF) cannot be accessed from outside (the MOVING platform) and vice versa. The only way to ensure a communication between the two applications is using JavaScript events. This event handling and triggering is managed by the `gvfapi.js`.

3.3 Responsive design

As originally described in Deliverable "D4.1: Definition of platform architecture and software development configuration", we are using the Bootstrap framework[12] in order to implement the responsive design making the platform easily and efficiently accessible via multiple types of devices, from desktop web clients to smartphones and tablets. For implementing the current version of the responsive design, the MOVING platform mock-ups were used as a starting point. They were developed with the Balsamiq Software[13] and described in detail in Deliverable "D1.1: User requirements and Specification of the use cases".

In accordance with these mock-ups, we organise the different functionalities in the same way in all the views of the application. Particularly, the faceted search is placed on the left, the visualisations in the middle and the Adaptive Training Support on the right of each responsive design view. Also, each view hosts on the upper

[12]http://getbootstrap.com, last accessed at 22.09.2017
[13]https://balsamiq.com, last accessed at 22.09.2017

part some navigation buttons (funding, research, project management, learning environment and community), and the button "My account". It is worthy to note that, in response to initial usability tests (described in the upcoming Deliverable "D1.2: Initial implementation of the user studies"), some parts of the mock-ups were not implemented because the corresponding functionalities overlap, while others were modified accordingly to the findings of these tests. Nevertheless, most mock-ups were implemented exactly as documented in Deliverable D1.1. More details regarding the implementation and the differences between the early version of the mock-ups and the implemented responsive design are given below.

1. For the "Research" view (D1.1, Sections 7.1 to 7.3):
   (a) For the "Faceted search" and "Sources" components (see Deliverable D1.1, Section 7.1), we implemented the dropdown menus by using multiple checkboxes for each single facet. As mentioned in Deliverable D1.1 this list shows possible facets at this stage of the project, which are not complete. An updated version will be implemented after month 24, when the updated version of the mock-ups will be available.
   (b) The "Search list" component, was implemented as described in the respective mock-ups.
   (c) The visualisations views (concept graph, tag cloud, top concepts, date mentions) were implemented as defined in the respective mock-ups.
   (d) For the Adaptive Training Support (ATS) widgets, we implemented the support for using the platform (see 1 in Figure 5(a)) and the adaptive training for the curriculum (see 2 in Figure 5(a)), as it is documented in all the mock-ups.
2. For the "Funding environment" view (D1.1, Section 7.4), the responsive design was implemented as it is in the mock-up by creating dropdown menus with multiple checkboxes.
3. For the "Community" view (D1.1, Section 7.5), we implemented the responsive design as showed in the mock-up by using dropdown menus with multiple checkboxes.
4. For the "Learning environment" view (D1.1, Section 7.6), we implemented the "Learning opportunities due to search queries" component and all the subsequent mock-ups that are described in the section. Some of the Learning environment components (e.g "Take a tour through the platform (Tutorials)" and "See all learning resources (MOOC, wiki)" components) are currently not fully specified in mock-ups, and will be implemented after month 24, when the updated version of the mock-ups will be available.

In Figure 5, we give an indicative example on how the mock-up was implemented in the responsive design regarding the "Community" view with the "Community" dropdown menu on the left side of the screen and the ATS widget of the right side of the screen.

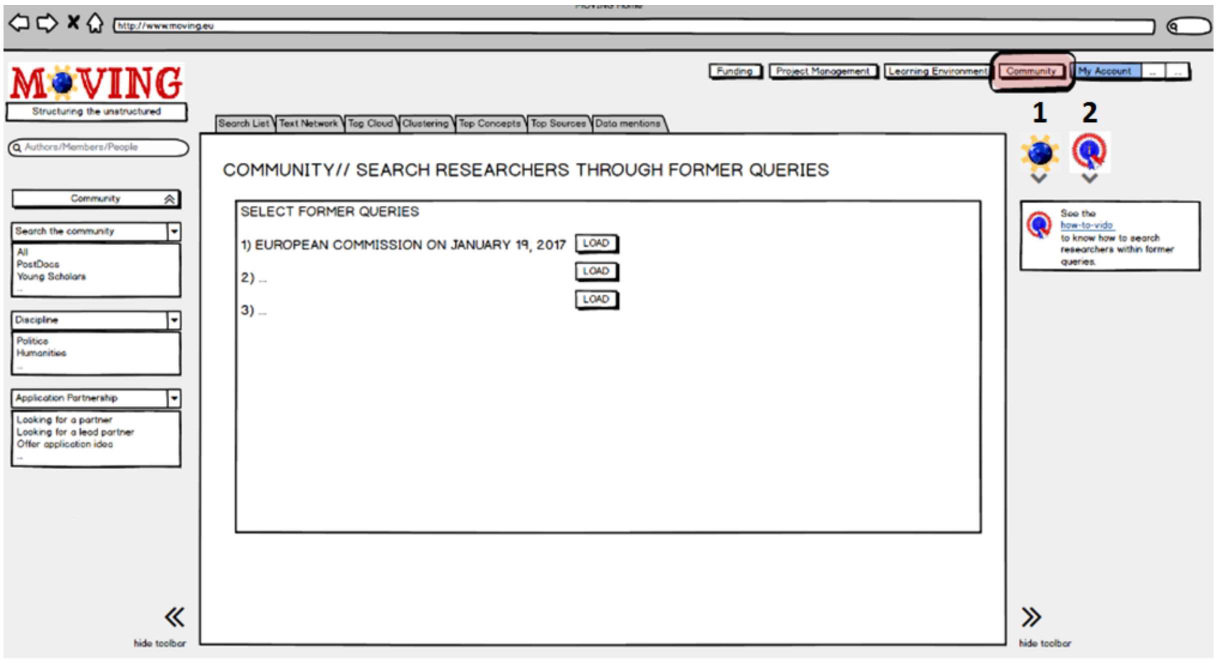
Figure 6, shows another example of an updated mock-up that was implemented in the responsive design. The user can select to watch a specific MOOC. For instance, selecting the 'MOOC1' link will result to viewing the MOOC in a desktop device as depicted in Figure 6(b).

All the aforementioned views can adapt with a multitude of different screen sizes, on which their layout is automatically changed based on the size and capabilities of the device. For example, on a PC screen, the users see the content in a three-column view as depicted in figure 7(a); on a mobile phone, the users can see content in a single-column view as depicted in Figure 7(b); and on a tablet they can see the same content but with the menus on the top of the screen, as shown in Figure 7(c).

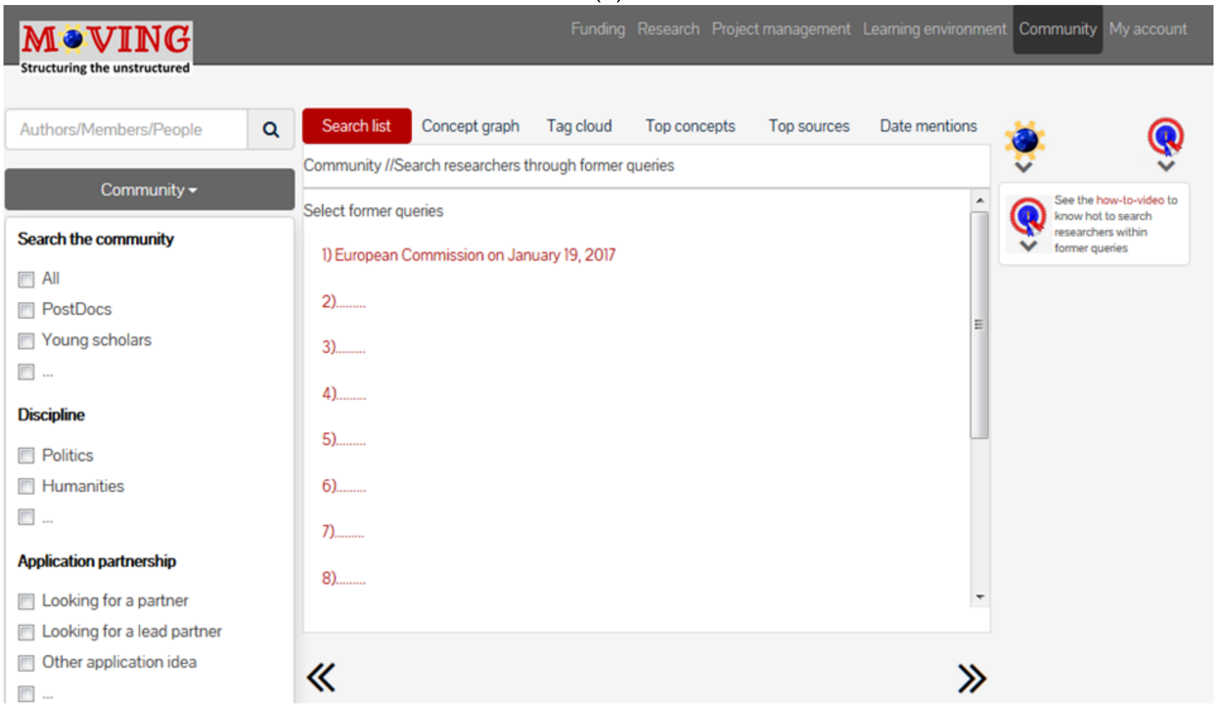
We employ this responsive design for all newly developed functions of the MOVING platform. Additionally, we are gradually introducing it for the preexisting platform functions of the eScience platform. At the time of writing, the responsive design is available for:

– the search frontend, including the graph visualisation;
– the landing page including the self-registration form;
– the onboarding wizard for newly registered platform users;
– the profile management screens;
– the user search;
– parts of the project management system: the project tree, the project information screen, the project settings screen and the Wiki module.

We plan to provide the responsive design for the complete web application during the remaining time frame of the project.

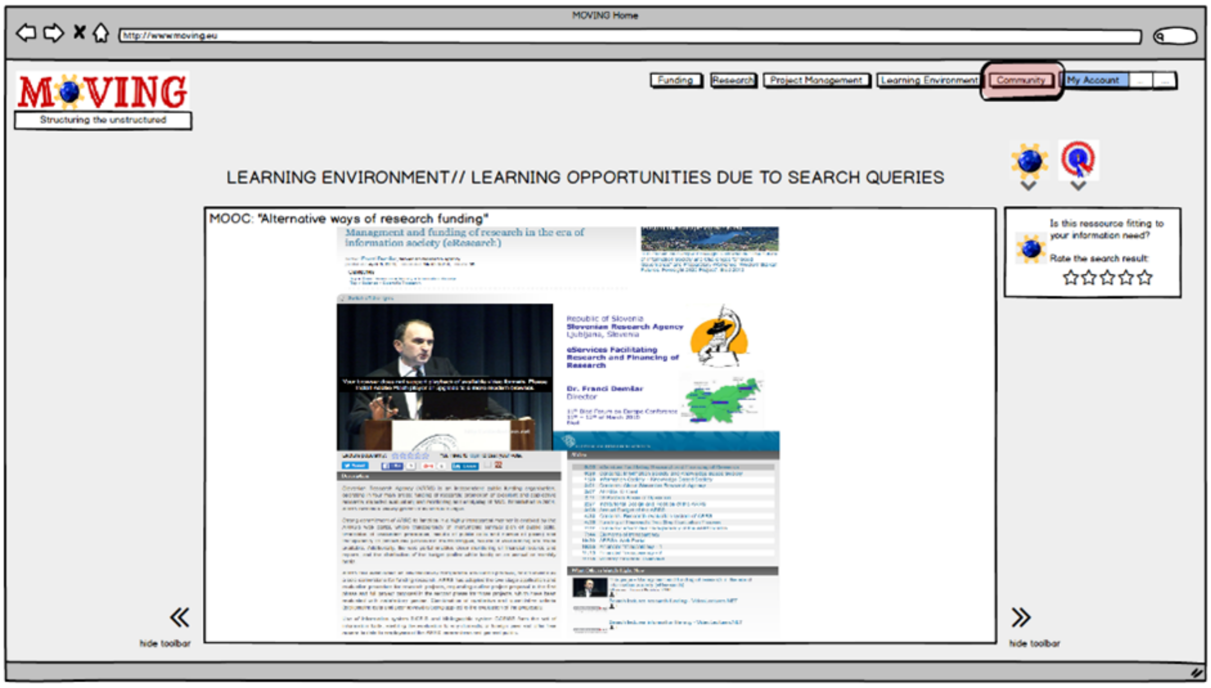


(a)

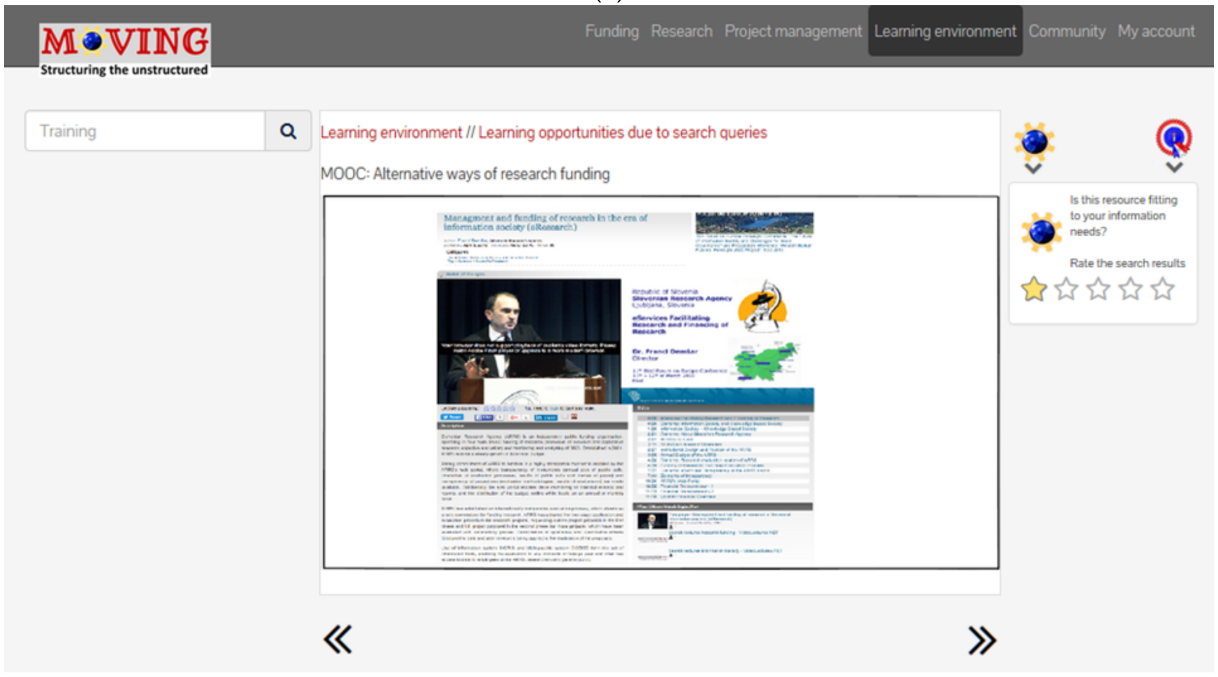


(b)

Figure 5: "Community" view comparison between the mock-up (a) and the implemented responsive design as shown in a desktop web browser (b).

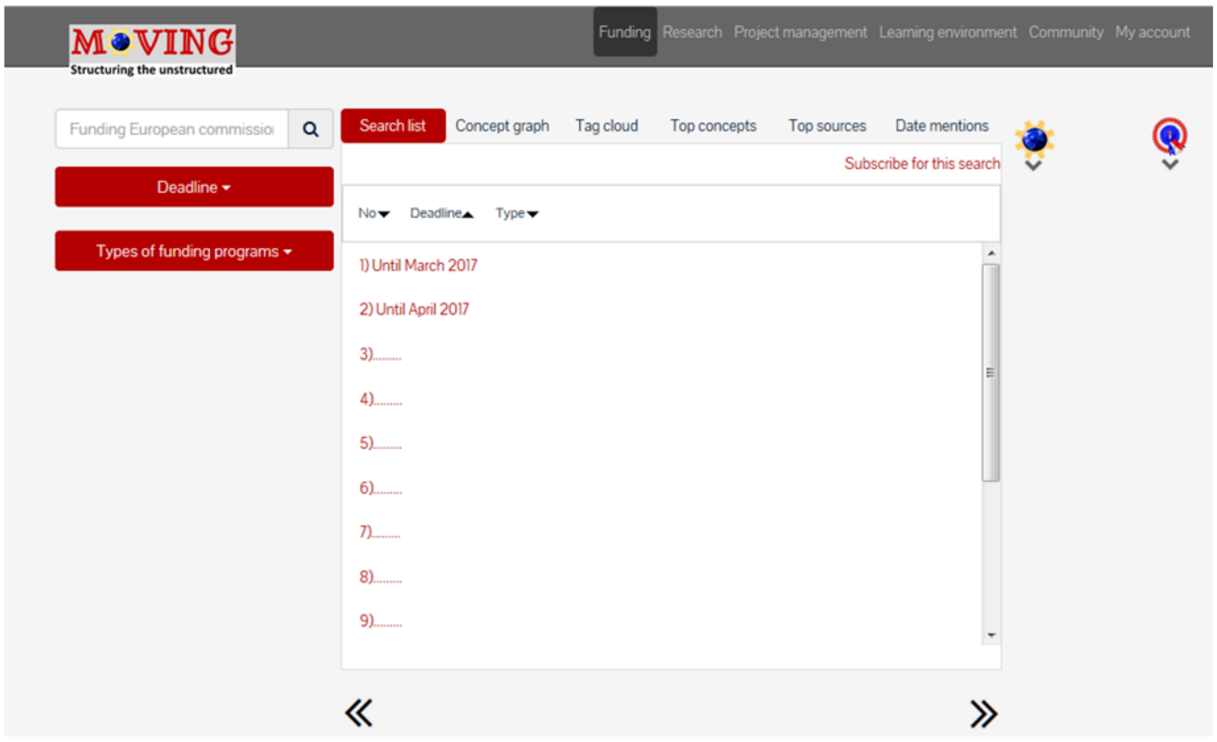

MOVING Home
http://www.moving.eu
Funding
Research
Project Management
Learning Environment
Community
My Account
Structuring the unstructured
LEARNING ENVIRONMENT// LEARNING OPPORTUNITIES DUE TO SEARCH QUERIES
MOOC: "Alternative ways of research funding"
Is this ressource fitting to your information need?
Rate the search result:
hide toolbar
hide toolbar

(a)

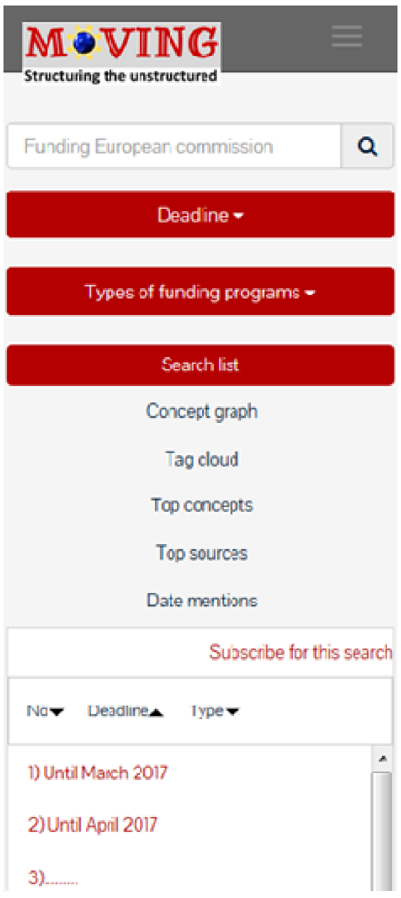

Funding
Research
Project management
Learning environment
Community
My account
Structuring the unstructured
Training
Learning environment // Learning opportunities due to search queries
MOOC: Alternative ways of research funding
Is this resource fitting to your information needs?
Rate the search results

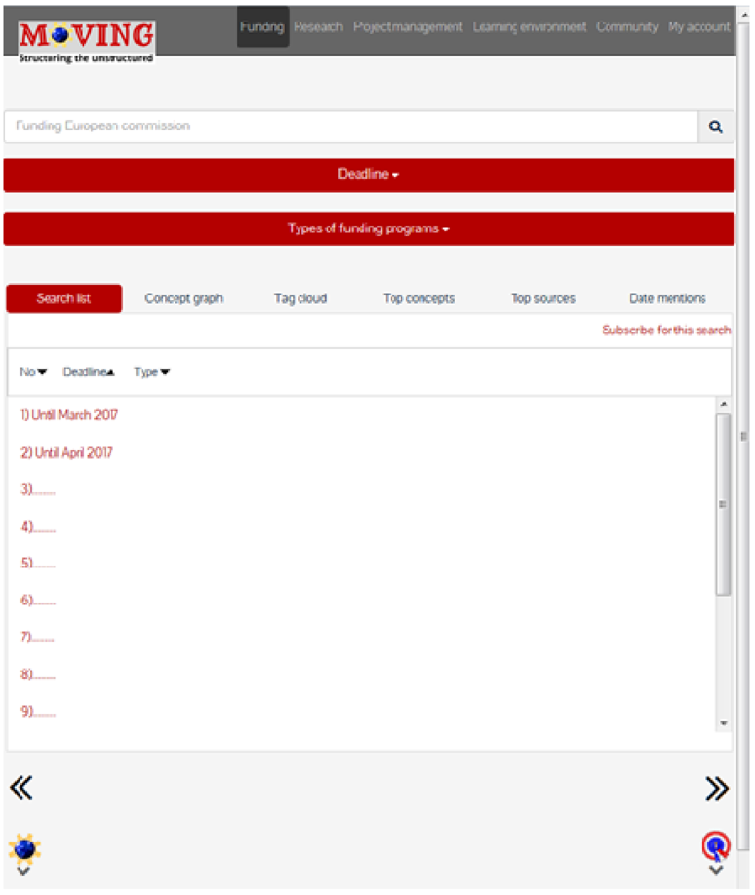
(b)

Figure 6: "MOOC" view comparison between the mock-up (a) and the implemented responsive design as shown in a desktop web browser (b).



(a)





(b) (c)

Figure 7: An example of the MOVING responsive design on a desktop pc or laptop (a), smartphone screen (b) and tablet (c).

4 Search engine

The MOVING platform needs to process huge amounts of text data coming from different data sources efficiently and effectively. In MOVING, we combine our own data sets, with existing data from the Linked Open Data cloud[14], a global space of structured and interlinked data. To this end, we implemented a variety of crawling and harvesting approaches (see Section 7). The MOVING platform contains also the following datasets:

1. Videolectures.net: the dataset consists of around 20,000 metadata records of educational videos with transcripts. The lectures are given by scholars and scientists at events like conferences, summer schools, workshops and science promotional events from many fields of science.

2. ZBW economics dataset(*MOVING_Data_WP3_11_ZBWEconomicsDataset* in Deliverable D6.2 “Data management plan”): metadata records of around 413,000 of economic scientific publications in English.

Moreover, the platform is currently integrating other data sources. For instance, GESIS Dataset *MOVING _Data_WP3_4_PublicationMetaData* (D6.2) which contains of around 2,8 million metadata records and 5.400 open access full texts. In order to handle these data sources efficiently and effectively, the MOVING's search engine needs to provide a scalable real-time search, support for multiple document types per index, different file formats and different programming languages. Thus, we used Elasticsearch for implementing the search engine of our MOVING platform. Elasticsearch allows adding custom functionalities to the index, and extensions like custom scoring methods can be integrated more efficiently. In Section 4.1, we describe how we integrated our novel ranking function (HCF-IDF) in Elasticsearch and present more details about the current index settings and mappings. Subsequently, in Section 4.2 we briefly describe the ranking models.

4.1 Elasticsearch setup

In order to provide a near instantaneous search experience over the huge amount of data in the MOVING platform, we followed the optimal configurations which have been recommended by Elasticsearch. Figure 8 shows a general overview of our Elasticsearch search setup. Our MOVING index is stored in one cluster. Below, we briefly introduce the clusters concept in Elasticsearch. Furthermore, we describe how we configured our MOVING cluster to allow scalability and increase the availability of our index.

MOVING cluster: Elasticsearch provides federated indexing and search capabilities through the cluster-nodes structure. One cluster contains one or more nodes (servers). As shown in the Figure 8, we configured our MOVING cluster to consist of only one node (on a server of the TU Dresden).

MOVING node: The node stores all our MOVING data. In the future, we can increase the number of nodes to scale sufficiently with our data volume.

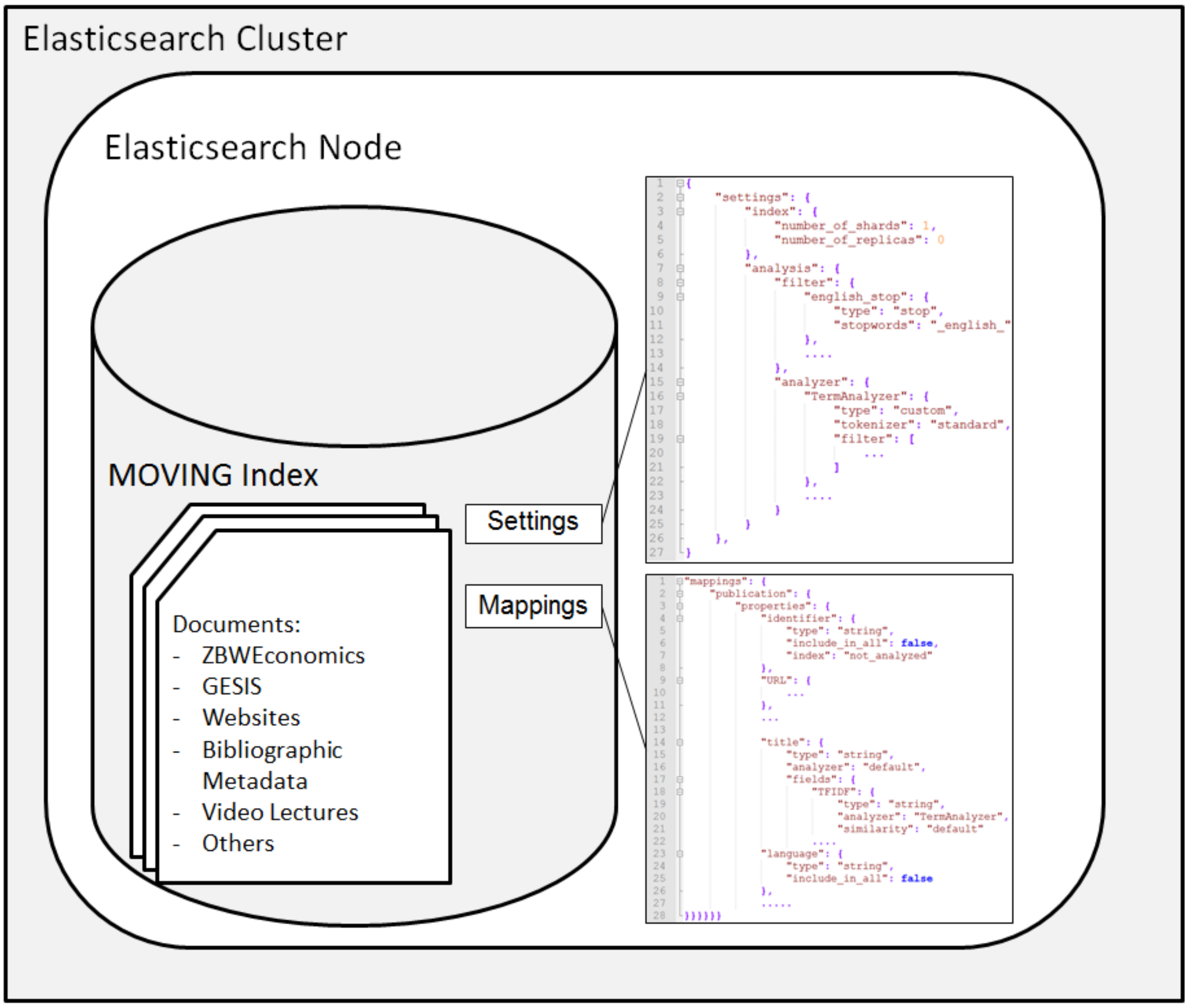
MOVING index: Our MOVING datasets are stored in an index inside the node. In Deliverable 4.1 “Definition of platform architecture and software development configuration”, we have presented more information about indexing the data in Elasticsearch, our tooling (elastify) and the main elements of an Elasticsearch Index (filter settings, analyzer settings, general settings and mappings). Below we describe the current index settings and mappings. In addition we present how we integrated out novel ranking function as a plugin to Elasticsearch.

The main elements of the index settings are:

Index shards. An Elasticsearch index may contain one or more shards. Each shard is a Lucene index which can hold up to $2,147,483,519$ documents [15]. We allocated our MOVING index to one shard because our current datasets volume is below this limit. The settings of an index describes how the document is analysed and stored. Elasticsearch not only provides numerous different built-in analysers but also offers the possibility to create custom analysers.

Index replicas. In order to ensure the availability of the platform, Elasticsearch offers a functionality of setting a replica to the index. In case of any fail-over, the replica index act as a primary index. During the development phase, we are setting the number replicas to zero. Listing 1 shows the current number of shards and replicas in our MOVING index settings.

[14] http://lod-cloud.net, last accessed at 21.09.2017
[15] https://www.elastic.co/guide/en/elasticsearch/reference/2.3/_basic_concepts.html, last accessed at 21.09.2017



```
{
    "settings": {
        "index": {
            "number_of_shards": 1,
            "number_of_replicas": 0
        },
        "analysis": {
            "filter": {
                "english_stop": {
                    "type": "stop",
                    "stopwords": "_english_"
                },
                ....
            },
            "analyzer": {
                "TermAnalyzer": {
                    "type": "custom",
                    "tokenizer": "standard",
                    "filter": [
                        ...
                    ]
                },
                ....
            }
        }
    },
}
```

```
"mappings": {
    "publication": {
        "properties": {
            "identifier": {
                "type": "string",
                "include_in_all": false,
                "index": "not_analyzed"
            },
            "URL": {
                ...
            },
            ...

            "title": {
                "type": "string",
                "analyzer": "default",
                "fields": {
                    "TFIDF": {
                        "type": "string",
                        "analyzer": "TermAnalyzer",
                        "similarity": "default"
                    ....
            "language": {
                "type": "string",
                "include_in_all": false
            },
            .....
}}}}}}
```

Figure 8: Elasticsearch setup in the MOVING platform

```
"index":
        "number_of_shards": 1,
        "number_of_replicas": 0
```

Listing 1: Index shards and replicas configurations

Filters. Various filters have been added to the MOVING index. These filters are utilised by the analysers to support the functionality of the ranking models (see Deliverable D3.1 "Technologies for MOVING data processing and visualisation v1.0"). Listing 2 shows the current filters which have been implemented in our MOVING platform.

```

        "analysis":
            "filter":
                "english_stop": ...,
                "english_possessive_stemmer": ...,
                "english_kstem": ...,
                "glue_together": ...,
                "alt_labels":
                    "type": synonym, "expand": false,
                    "synonyms_path": analysis/altLabels.txt
                "pref_labels":
                    "type": keep, "keep_words_path": analysis/prefLabels.txt
                "prefLabels_to_descId":
                    "type": synonym, "expand": false,
                    "synonyms_path": analysis/prefLabel2descId.txt
                "spread2root":
                    "type": synonym, "expand": false,
                    "synonyms_path": analysis/spread2root.txt
```

Listing 2: MOVING index filters configuration

The main filters are presented in the following items:

– ”*alt_labels*, *pref_labels*” : STW thesaurus [16] contains information about economics concepts. Some concepts (*alt_labels*) have a preferred synonyms (*pref_labels*). In our Elasticsearch configurations, we added a custom filter to add the synonyms of the concepts using some external files.

– ”*english_stop*” [17] is used to filter english stop words (e.g. ”the”).

– ”*english_possessive_stemmer*”,”*english_kstem*”: Arithmatic stemmers apply different rules to return a word to its root form. For instance, converting the plural form of a word like ”Tax Offices” to the its singular form ”Tax Office”.

– *Spread2root*: Each thesaurus consists of a comprehensive list of subjects concerning which information may be retrieved by using the proper key terms. The list of subjects are usually following a hierarchy. Some ranking models (e.g. HCF-IDF) use the hierarchical information to improve the search results. The MOVING platform can handle different kind of thesauruses (e.g. STW, MeSH [18] and FIV[19]).

Analysers. Elasticsearch provides a wide range of built-in analysers for handling the index text (see Deliverable 4.1). An analyzer may consist of many filters. We developed custom analyzers, called “ConceptAnalyzer” and “SpreadingActivationAnalyzer”, in order to pre process different thesauruses and the indexed documents. These analysers are used by our HCF-IDF plugin to rank the documents based on their relevance to the user query (more details below). Listing 3 shows the current analyzers configurations in our MOVING index.

```

            "analyzer":
                "TermAnalyzer":
                    "type": custom, "tokenizer": standard,
                    "filter": [english_possessive_stemmer, lowercase, english_stop, english_kstem]
                "ConceptAnalyzer":
                    "type": custom, "tokenizer": standard,
                    "filter": [english_possessive_stemmer, lowercase, english_stop, english_kstem, glue_together, alt_labels,
                        ↪ pref_labels]
                "SpreadingActivationAnalyzer":
                    "type": custom, tokenizer: standard,
                    "filter": [english_possessive_stemmer, lowercase, english_stop, english_kstem, glue_together, alt_labels,
                        ↪ pref_labels, prefLabels_to_descId, spread2root]
```

Listing 3: MOVING index analysers

In Listing 4, our index mappings reflect our common data model v1.0 (see Deliverable D3.1 “Technologies for MOVING data processing and visualisation v1.0”). Particularly, they define how each property (e.g. full text or title) of the document can be retrieved. It is possible for one property to define multiple fields of which each has its own type, analyser and similarity module. Thus, we created a field using BM25, TF-IDF and

[16] http://zbw.eu/stw/version/latest/about, last accessed at 21.09.2017
[17] https://www.elastic.co/guide/en/elasticsearch/guide/current/using-stopwords.html, last accessed at 21.09.2017
[18] https://www.ncbi.nlm.nih.gov/mesh, last accessed at 21.09.2017
[19] http://www.fiv-iblk.de, last accessed at 21.09.2017

other ranking modules. This way we can create the features for each property by retrieving documents using the corresponding field.

```
    "mappings": {
        "publication": {
            "properties": {
                "identifier": {....},
                "URL": {....},
                "documentURLs": {....},

                "title": {
                    ...
                    "fields": {
                        "TFIDF": {
                            "type": "string",
                            "analyzer": "TermAnalyzer",
                            "similarity": "default"
                        },
                        "BM25": {....},
                        "CFIDF": {....},
                        "BM25C": {....},
                        "HFIDF": {
                                "type": "string",
                            "analyzer":"SpreadingActivationAnalyzer",
                            "similarity": "semanticsimilarity"},
                        "HFBM25": {....},
                    }
                },
                "abstract": {....},
                "fulltext": {....},
                "authors": {....},
                ....
                "keywords": {....},
                "references": {....}
}}}
```

Listing 4: MOVING index mappings

4.2 Ranking algorithms

Elasticsearch enables us to retrieve a number of relevant documents with respect to the user query. Elasticsearch computes the similarity scores for each query-document pair. The similarity scores are used to rank the documents based on their relevance to the user query. Different ranking models have been implemented in the MOVING platform to rank the search results. More details are presented in Deliverable D3.1 "Technologies for MOVING data processing and visualisation v1.0", D3.2 "Technologies for MOVING data processing and visualisation v2.0" and D2.1 "Initial conceptual framework, curricula and technical prototypes for adaptive training support".

HCF-IDF plugin As presented in Listings 3 and 4, a new analyzer (namely "SpreadingActivationAnalyzer") and a new similarity function (namely "SemanticSimilarity") have been developed in order to rank the document based on the novel ranking algorithm HCF-IDF.

5 Adaptive training support (ATS)

The adaptive training support (ATS) visualises information about the use of features collected by the MOVING platform. It presents the information to the user to motivate them to explore the provided features and reflect about the usage behaviour.

Currently, the following features have been defined to be used for adaptive training support:

– **Basic Search** The basic search feature involves entering some keywords and submitting the search request via the standard search query button.

– **Faceted Search** The faceted search feature involves entering at least one of the search fields in the advanced search view, e.g. the name of the author. The user submits the search via the search query button in the advanced search view.

- **Result List** The result list feature shows the search results as a list. This happens when the user clicks on the 'Results' button in the search results view.
- **Concept Graph** The concept graph feature displays the search results with the concept graph visualisation. This happens when the user clicks on the 'Concept Graph' button in the search results view (see Section 3.2).

The ATS engine, the central component of the adaptive training support system, visualises the feature use of the MOVING platform's user community. Section 5.1 describes how the ATS engine creates information about the feature use and how it presents the latter in the MOVING web application. Section 5.2 describes how the adaptive training support makes users aware of the available features and what strategies are applied to motivate users to explore them.

5.1 Use of features

The ATS engine gathers information about the use of features in a semi-automatic way which involves:

- used feature detection in the interaction log;
- extraction and storage of features used
- continuous update of the features used

In the following we describe how the detection, extraction, storage and continuous update of used feature is done in the ATS engine.

Feature detection in the interaction log Feature detection in the ATS engine is based on the interaction data captured from the users of the MOVING platform. See Section 6 for more information about how WevQuery logs events of users while they interact with the web interface of the MOVING platform. The event sequences for the basic search, faceted search, result list and concept graph features were defined with the WevQuery's Web Interface:

Definition of 'Basic search' and 'Faceted search' features The event sequence for basic and faceted search consists of a two event sequence pattern, a `mouseup` event on the search button followed by a submit event. This pattern is suitable for basic and faceted search submits as well. The distinction between which type of search was actually performed is made later during the extraction of event data.

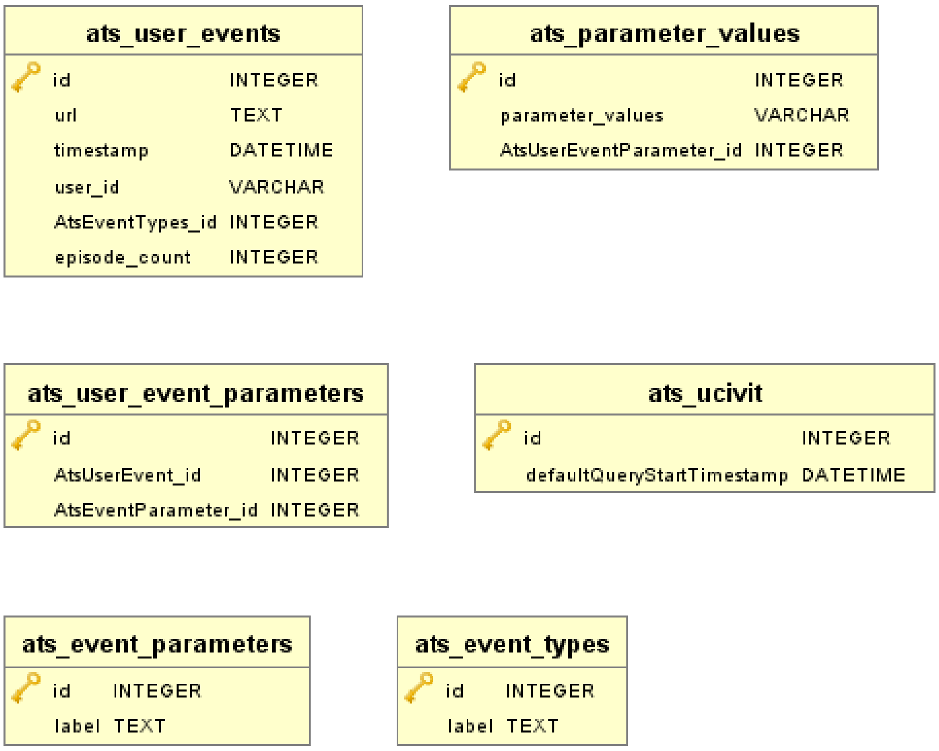
Definition of 'Result list' feature The result list feature is triggered when the user clicks on the Results button in the search results view.

Definition of 'Concept graph' feature The concept graph feature is triggered when the user clicks on the Concept Graph button in the search results view.

Feature extraction and storage After saving the definitions, the ATS engine calls the interface for each definition, processes the event data in the response, extracts and saves information about feature use in the ATS database.

ATS Engine database Figure 9 shows the database schema of feature use. Table `ats_user_events` stores data on the detected features which includes the following properties:

- **URL** The web URL at the time the feature was used;
- **timestamp** The date/time when the feature was used;
- **user_id** The id of the user who used the feature;
- **AtsEventTypes_id** The type of the feature which was used;
- **episode_count** The number of episode the feature usage belongs to. WevQuery defines episodes as the interaction from a single user that happen in a Web page without a *noticeable* interruption, see Section 6 for details.



ats_user_events	
id	INTEGER
url	TEXT
timestamp	DATETIME
user_id	VARCHAR
AtsEventTypes_id	INTEGER
episode_count	INTEGER

ats_parameter_values	
id	INTEGER
parameter_values	VARCHAR
AtsUserEventParameter_id	INTEGER

ats_user_event_parameters	
id	INTEGER
AtsUserEvent_id	INTEGER
AtsEventParameter_id	INTEGER

ats_ucivit	
id	INTEGER
defaultQueryStartTimestamp	DATETIME

ats_event_parameters	
id	INTEGER
label	TEXT

ats_event_types	
id	INTEGER
label	TEXT

Figure 9: ATS: Database schema of feature use

Table `ats_event_types` contains the list of features described by their label:

- **label** The feature label, e.g. 'Concept Graph'

Table `ats_event_parameters` contains the list of parameters which may belong to a feature.

- **label** The parameter label, e.g. 'advanced_query_title'

Table `ats_user_event_parameter` links the features used with their parameters.

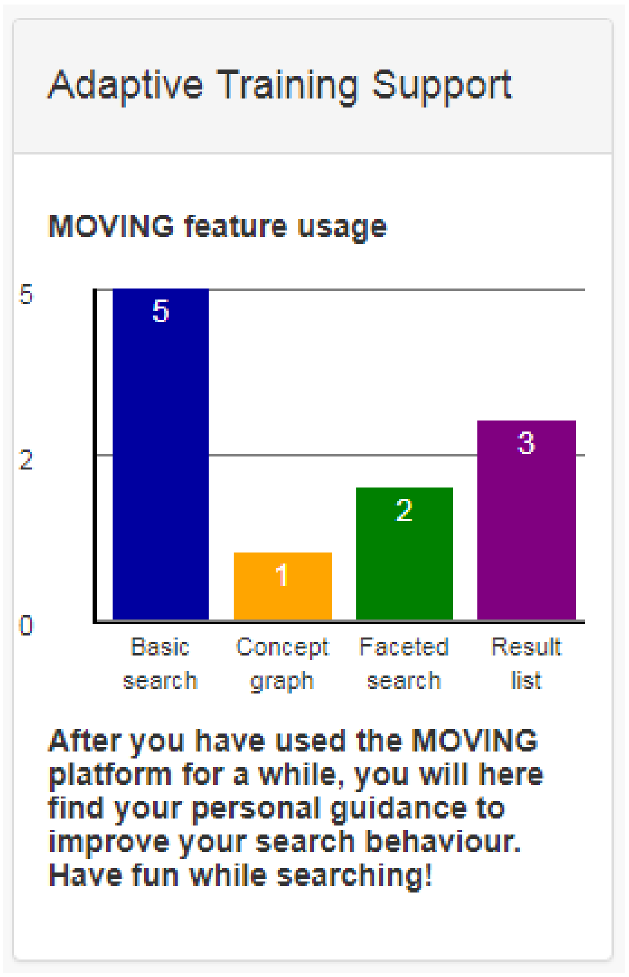
- **AtsUserEvent_id** References the features used from table ats_user_events
- **AtsEventParameter_id** References a parameter from table ats_event_parameters

Table `ats_parameter_values` stores the parameter value for a feature.

- **parameter_values** The parameter value

Continuous update The ATS engine continuously triggers an update of feature use for all users on the MOVING platform. Each update involves i) the feature detection from the interaction log ii) the extraction and storage of features in the ATS database iii) the update of the timestamp used as start timestamp for feature detection and iv) the schedule of the next update. The ATS engine maintains the start timestamp for feature detection. At the end of an update this timestamp is shifted forward to the timestamp of the last detected feature. This ensures that only the events in the interaction log starting from the one after the last detected feature are processed during an update. The ATS engine also maintains a parameter which defines the time interval between consecutive updates of the feature use on the MOVING platform. After finishing an update the engine schedules the next update with the time interval of this parameter. This way the feature use for each user on the MOVING platform is kept up to date.

Visual feedback in the ATS Widget The ATS is presented as a widget on the right hand side of the MOVING platform's web interface. The feature usage is presented in the top area of the widget. It is a two-dimensional chart showing the use for each feature in form of colored bars. Each bar is labelled with the absolute number of occurences. Figure 10 shows the screenshot of the MOVING web interface with the ATS widget showing 'basic search' as the most used feature in this case.



After you have used the MOVING platform for a while, you will here find your personal guidance to improve your search behaviour. Have fun while searching!

Figure 10: ATS Widget: Basic search is the most used feature

5.2 Reflection guidance

The reflection guidance presents a reflective prompt in form of a reflective question or sentence starter to stimulate reflection on the search behaviour. The goals of reflection guidance are:

– motivate users to reflect on past feature usage (reflection amplifier);

– motivate users to use another feature (reflection intervention).

The prompts presented to the user depend on the user's experience with the search features. The ATS takes information about the feature used as a measure of experience, as described in Section 5.1. The ATS exploits this information to decide which prompt is presented next to the user. When the user starts using the MOVING platform the ATS has no information about the user's experience. The user should get familiar with the features and should have the opportunity to draw a first impression. Reflection guidance in the ATS widget at this stage (start up stage) does not present any prompt. Rather it shows the user that personal guidance will show up in this area of the ATS widget sometime in the future. Figure 10 shows how the ATS widget looks like at this stage. Reflection guidance starts when the user accomplishes to use the search features a certain amount of time. The ATS widget shows the prompt and a text field where the user can enter the answer. When the user submits the answer, the prompt and answer field disappears. The reflection guidance does not show the next prompt immediately. It waits a certain amount of time until the next prompt is selected and shown to the user. The episode count is used to decide when to show the next prompt. The default is one episode of not showing any prompt. This way it is avoided to annoy the user with too many prompts. The reflection guidance model tracks feature use and the answers given to the presented prompts. Based on this information it decides if it keeps presenting prompts from the current category or if it moves on to the next one. After the start up stage three more stages in the reflection guidance model follow. In each stage the reflection guidance model uses one or more categories to select prompts. The categories between each stage differ and are adjusted to the experience the user should have at this stage. The prompt categories in the different stages:

– **first stage** prompts which features are less or not used, and also about the benefit and/or satisfaction of a specific feature.

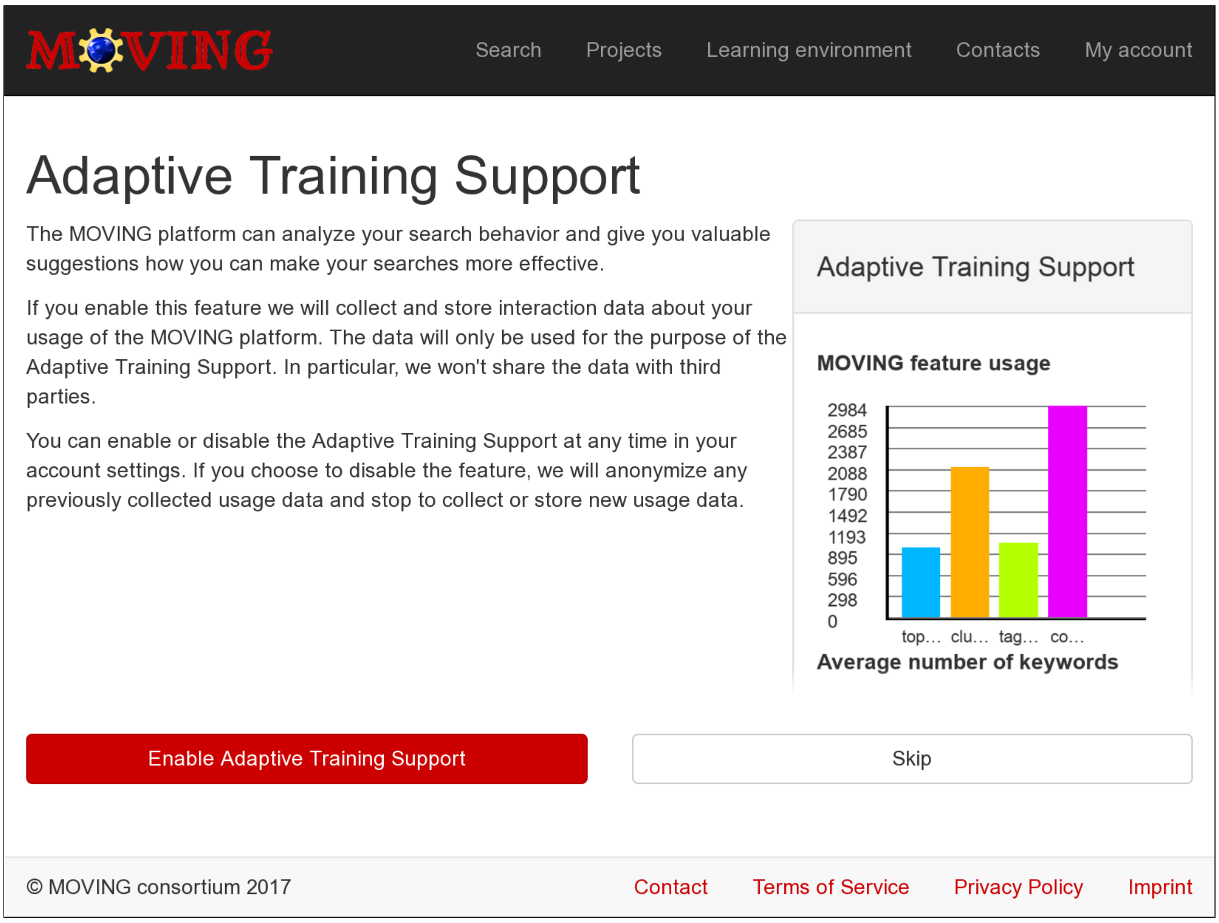
– **second stage** prompts about the feature mostly used in the platform, prompts about the reason why these features which are mostly used or less used, and reflection.

– **third stage** prompts about most beneficial/satisfactory features, and skill/performance increase, behaviour changes.

The prompts in the first stage are easy to answer and should keep user's motivation high to further interact with the ATS widget. The switch from the first into the second stage happens when the prompts for each of the categories in the first stage has been answered and feature use exceeds a certain threshold. In the second stage the prompts are aligned to the experience user's have collected so far. The effort of answering the prompts is higher than in the first stage. The switch to the third stage depends as before on answering the prompts and feature use. In the third stage the most challenging questions in terms of reflection are presented to the user. These questions are not only about feature use. E.g., they address the question whether the user has observed a change in his search behaviour influenced by the ATS widget.

6 User interaction tracking and dashboard

To comply with the European data protection law the user interaction tracking is turned off by default. When a new user signs in to MOVING platform for the first time, he is asked if he wants to allow the data collection process. Only when he agrees with the collection of his user interaction data, the tracking code is enabled. Figure 11 shows a screenshot of the user interface used for the opt-in to the user interaction tracking.



Adaptive Training Support

The MOVING platform can analyze your search behavior and give you valuable suggestions how you can make your searches more effective.

If you enable this feature we will collect and store interaction data about your usage of the MOVING platform. The data will only be used for the purpose of the Adaptive Training Support. In particular, we won't share the data with third parties.

You can enable or disable the Adaptive Training Support at any time in your account settings. If you choose to disable the feature, we will anonymize any previously collected usage data and stop to collect or store new usage data.

Figure 11: Opt-in for the adaptive training support as part of the MOVING platform onboarding process

The user can enable or disable the user interaction tracking at any time via the profile settings. If he chooses to disable the tracking, all the data collected about his usage behaviour is anonymized. This is achieved by deleting the tracking identifier in his user profile. This identifier is chosen randomly. Once it is removed one cannot link the collected tracking data to its original user profile. On the other hand, the behavioural data is not lost and we can still use the data to generate suggestions for other users.

When the capture of interaction data is enabled, various interaction events are collected from the Web application. These events provide enough context about the interaction so the information can be used to support the functionalities of the ATS (see Section 5) as well as the analysis of the use of the MOVING platform. Details about the interaction events, and the technology employed to capture the interaction from the users can be found in the Section 3.5.1 of Deliverable D4.1 “Definition of platform architecture and software development configuration”.

6.1 WevQuery

The interaction data captured from the users has the potential to provide rich information about their use of the MOVING platform. Unfortunately, the analysis of such low-level logs requires a high degree of expertise and domain specific knowledge, which the designers of systems which exploit these data might not have. As a way to ease the access and analysis of the captured data, an interaction dashboard has been implemented. Other MOVING components, such the ATS, can employ this dashboard to retrieve interaction information about the users. This way recommendations can be tailored, and individual user’s progress throughout their knowledge acquisition can be tracked. The dashboard is named WevQuery[20], which stands for Web Event Query Tool, and allows designers without knowledge of databases or programming skills to build queries to retrieve information about behaviours exhibited on the web application (Apaolaza & Vigo, 2017).

The analysis of low-level interaction logs is made accessible by providing a graphical tool to design queries interactively. These queries are represented as sequences of events that are defined using interactive drag-and-drop functionalities on a Web application. The queries can include a number of user interface events including scroll change events and mouse interaction. Temporal relations between the events can also be incorporated into the query: e.g. one can define the minimum time elapsed between two consecutive events. The system then translates this graphical representation into a query and searches for patterns of events that match the defined sequence.

This way WevQuery supports the extraction of specific interaction conditions. For example, the ATS might need the number of times that a specific feature was used in the platform. However, just clicking on a feature might not indicate that the feature was actually used. Instead, a more complex indicator can be designed, showing when the user actually interacted with that feature (e.g. the use of the “search” function can be defined as clicking on search, and interacting with the results afterwards). Designers can also design hypothesis about user interaction and test them against the captured interaction data. This way they can check if their expectations about the interaction are correct.

Figure 12 shows the general architecture of WevQuery, which consists of two main components: first, an interface with an interactive Web application that allows designers to graphically define their queries or hypotheses; second, the back-end translates these graphically designed hypotheses into queries that are run against a database and produces a comprehensive report about the formulated hypotheses.

Figure 13 is a screen capture of WevQuery’s graphical interface and shows the functionalities designers can use to construct the queries that test their hypotheses. Queries are defined as a sequence of events and, when executed, look for patterns of events matching that sequence. Temporal restrictions can be included in order to establish intervals of time between events that match the query. The graphical interface is dynamically built from an XML Schema (see Appendix A) that defines the grammar of the queries, which ensures the resulting query complies with the requirements of the system. Possible values for other fields are also retrieved from this schema, such as the attributes of the node elements that can be used to further specify an event. Such queries are stored as XML files that conform to the mentioned schema so that designers can reuse and share them at any time. An example of the resulting XML file can be seen in Listing 21. Then the query in the XML file is transformed into a MapReduce query. The XML schema also serves to check the validity of the produced XML file, so WevQuery can transform them into the MapReduce query. MapReduce (Dean & Ghemawat, 2008) is a programming paradigm designed to handle large datasets. MapReduce makes use of two functions: the *map* function splits the data into subsets and then the *reduce* function processes each subset independently. We have chosen this paradigm to ease the processing of large amounts of interaction data and make it scalable and suitable for distributed systems. This paradigm is also particularly suited for WevQuery, as interaction data from individual user episodes is processed independently. The MapReduce query is then run against the

[20] Available at https://github.com/aapaolaza/WevQuery

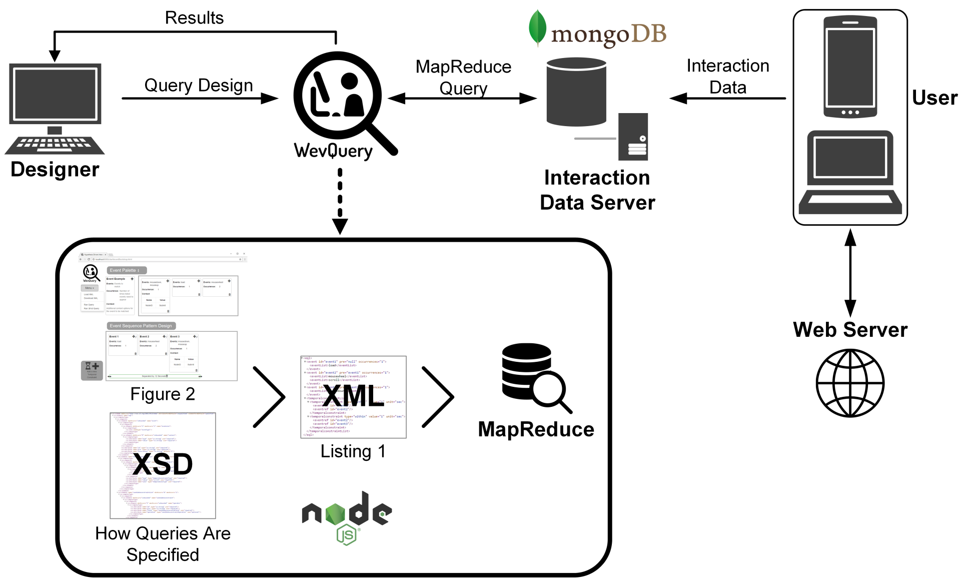


Figure 12: Architecture of WevQuery

Interaction Data Server, which stores user interface events in a MongoDB[21] database. WevQuery runs on Node.js[22], which is used to host the Web application serving the graphical interface as well as to execute the designed queries and connect to the MongoDB database.

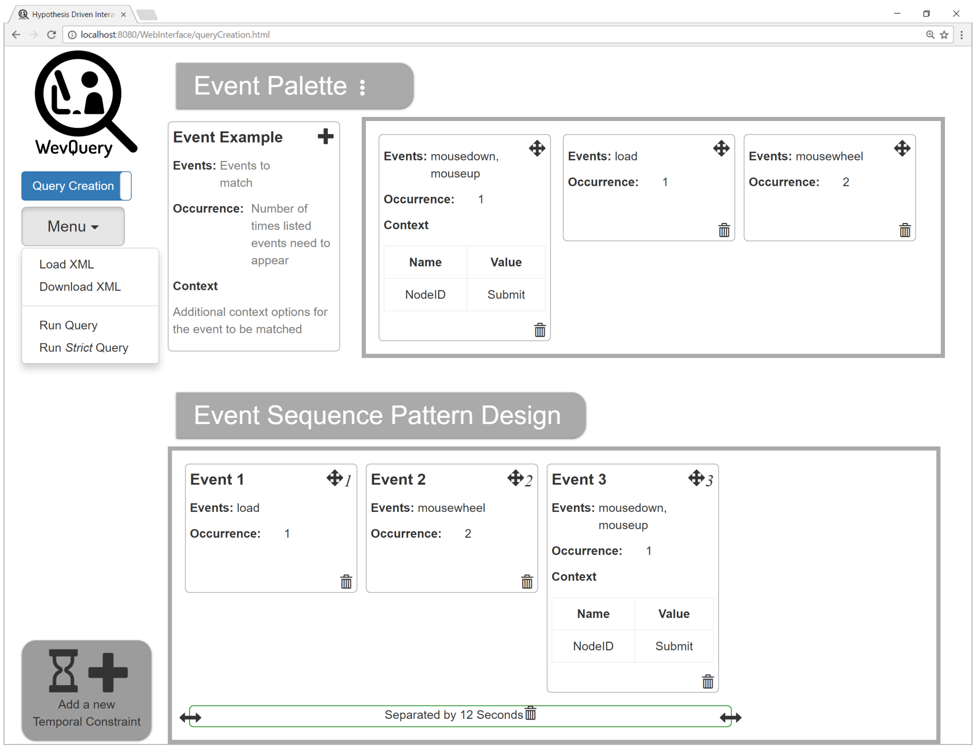
Defining the Queries Queries are formally specified through a set of rules that define the values and attributes of user interface events and the relations between different events. Relying on a formal structure makes possible to automatically transform the queries into scalable database queries. Such formalisms were defined using an XML Schema that was inspired by works that identify requirements of complex models in event-based systems (Scherp, Franz, Saathoff, & Staab, 2009). The use of XML provides multiple advantages: it can be easily validated against the defined structure, it is human readable, it can be easily extended with additional query features, and its use is suitable for programmatically deriving MapReduce queries. As above-mentioned the graphical interface shown in Figure 13 also uses the XML Schema to generate the graphical elements that allow to define interaction patterns on-the-fly based on the current definition of the query.

Taking into account the sequential nature of the captured interaction events, our current schema implements a subset of the possible relations between user interface events (Allen, 1983): *precedes* and *preceded by*. These relations indicate either the following or the preceding event within a set of events arranged in a particular order and discarding overlaps.

Interaction events are captured synchronously in a strictly ordered manner. Overlaps between these interaction events are not possible as interaction events are pinpointed in time and they are atomic. An example of a query designed with WevQuery can be seen in Listing 21. The order between the events is set by declaring each events' predecessor using the `pre` attribute while remaining attributes store values of the query. The various event elements describe the pattern to search: a single `load` event, followed by a single occurrence of either `mousewheel` or `scroll`, ending on a single `mousedown`. temporalconstraintList element contains two temporalconstraint that restrict the query by defining temporal relationships between events: these constraints ensure that the time elapsed from event1 to event2 and from event2 to event3 is less than 1 second. This is conveyed by the within value of the type attribute, which sets the scope of the interval, and specified by the value and unit attributes.

Defining Queries through WevQuery's Web Interface The graphical user interface shown in Figure 13 enables designers to build queries. This interactive Web application supports the design of the queries, allowing designers to drag and drop the event elements, and automatically showing the available values for the attributes of the events. The queries are composed of a sequence of ordered events and each event in the sequence can match one or more types of interaction events. For example, a particular event in the sequence can be set

[21]https://www.mongodb.com/, last accessed at 21.09.2017
[22]https://nodejs.org/, last accessed at 21.09.2017



Event Example

Events: Events to match

Occurrence: Number of times listed events need to appear

Context

Additional context options for the event to be matched

Events: mousedown, mouseup

Occurrence: 1

Context

Name	Value
NodeID	Submit

Events: load

Occurrence: 1

Events: mousewheel

Occurrence: 2

Event 1 — 1

Events: load

Occurrence: 1

Event 2 — 2

Events: mousewheel

Occurrence: 2

Event 3 — 3

Events: mousedown, mouseup

Occurrence: 1

Context

Name	Value
NodeID	Submit

Separated by 12 Seconds

Figure 13: Screenshot of the WevQuery Web application

to match either a `mousedown` (describes the action of clicking the mouse) or a `mousewheel` (describes the interaction with the scroll wheel of the mouse). In the case of the first event from the Event Palette (see below in Figure 13), it matches two different event types: `mousedown` and `mouseup`. The graphical interface consists of several modules including an event palette, a widget to define sequences and components to establish the temporal relationships between events.

The Event Palette This widget displays the user events that can be selected when defining a query. When the designer presses on the *plus* sign in the “Event Example” box (under the Event Palette header in Figure 13), the event template creation dialogue is shown as depicted in Figure 14a. One or many event types can be selected, which are loaded from the XML Schema. The number of times an event type needs to be matched can be set in the “occurrence” field.

For example, a `mousedown` event with an occurrence value of 2, is equivalent to a sequence of two independent `mousedown` events with occurrence value of 1. The context for the match can also be set by specifying the element of the user interface that triggered such event, using the attributes of the Document Object Model (DOM) node to do so. It can also set the scope of the Web page where the event took place by specifying its URL(s). Once the event is defined it is added to the palette so that it can then be dragged into the widget that allows the specification of event sequences – see below.

Designing Patterns of Event Sequences Events created in the Event Palette can be dragged and dropped into the Event Sequence Pattern Design area (bottom panel in Figure 13), using the *move* icon at the top-right of the element that represents an event. The position of events in the list determines the order of the sequence which is conveyed by a number located next to the mentioned *move* icon. The resulting query consists of a sequence of events WevQuery uses to look for patterns that match the sequence. Events can also be discarded by clicking on the *bin* icon located at the bottom-right corner of each event.

Defining Temporal Constraints The addition of temporal constraints allows designers to set time intervals between matched events. If not specified, the query will ignore the time elapsed between events. When clicking on the “Add a new Temporal Constraint” button, the dialogue that shows in Figure 14b pops up, allowing designers to establish the following temporal aspects:

```xml
<eql>
  <event id="event1" pre="null"
      occurrences="1">
    <eventList>load</eventList>
  </event>
  <event id="event2" pre="event1"
      occurrences="1">
    <eventList>mousewheel</eventList>
    <eventList>scroll</eventList>
  </event>
    <event id="event3" pre="event2"
        occurrences="1">
    <eventList>mousedown</eventList>
  </event>
  <temporalconstraintList>
    <temporalconstraint type="within"
        value="1" unit="sec">
      <eventref id="event1"/>
      <eventref id="event2"/>
    </temporalconstraint>
    <temporalconstraint type="within"
        value="1" unit="sec">
      <eventref id="event2"/>
      <eventref id="event3"/>
    </temporalconstraint>
  </temporalconstraintList>
</eql>
```

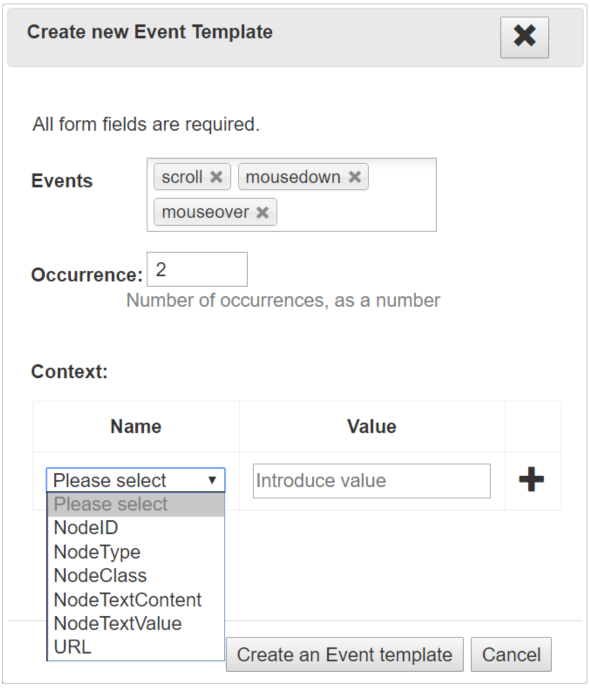
Listing 5: Example of query created with WevQuery

```json
"_id" :{
  "userID" :"2HZhjN5yQjAC",
  "url" :"http://www.cs.manchester.ac.uk/",
  "episodeCounter" :1
 },
"value" :{
  "xmlQuery" :[
   [{
     "event" :"load",
     "timestamp" :"2016-04-13,17:13:53",
     "timestampms" :1460564033927,
     "htmlSize" :"960x1054",
     "resolution" :"960x600",
     "size" :"960x431",
    },
    {
     "event" :"scroll",
     "timestamp" :"2016-04-13,17:13:54",
     "timestampms" :1460564034166,
    },
    {
     "event" :"mousedown",
     "timestamp" :"2016-04-13,17:13:54",
     "timestampms" :1460564034977,
     "mouseCoordinates" :{
       "coordX" :361,
       "coordY" :294,
       "offsetX" :22,
       "offsetY" :8
     },
     "nodeInfo" :{
       "nodeDom" :"id(\"Main\")/DIV[1]/A[2]",
       "nodeLink" :"/research/",
       "nodeText" :"Research",
       "nodeType" :"A",
       "nodeTextContent" :"Research",
       "nodeTextValue" :"undefined"
     }
    }]
  ],
  "isQueryStrict" :false
 }
```

Listing 6: JSON report generated as a result of a query

- **Relation** determines if the temporal distance between selected events has to be under (within) or above (separated by) the indicated threshold.
- **Events** allows the designer to select the two events affected by the temporal constraint. When one of the buttons is pressed, the dialogue temporarily disappears so that the user can select the events in the Event Sequence Pattern Design area (bottom panel in Figure 13).
- **Duration** and **Unit** determine the temporal distance, and the unit of time that designers wish to establish.

Once temporal constraints are defined, the length of the bar that conveys the scope of temporal constraints can be dragged and modified.


Create new Event Template

All form fields are required.

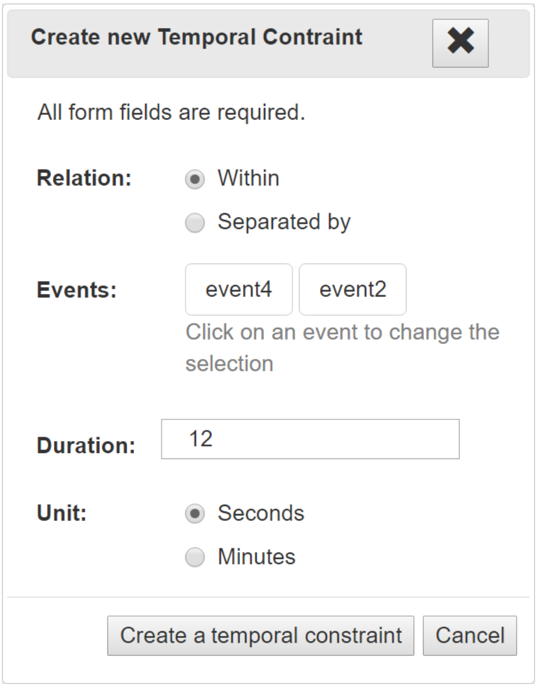
Events: scroll ✖ mousedown ✖ mouseover ✖

Occurrence: 2
Number of occurrences, as a number

Context:

Name	Value
Please select	Introduce value

(a) New Event dialogue


Create new Temporal Contraint

All form fields are required.

Relation: [● Within] ○ Separated by

Events: event4 event2
Click on an event to change the selection

Duration: 12

Unit: [● Seconds] ○ Minutes

(b) New Temporal Constraint dialogue

Figure 14: Additional dialogues of the WevQuery interface

File Menu The file menu at the top left corner of the WevQuery interface (see Figure 13) allows designers to operate with the designed query: once the query is defined, the designer can click on "Run Query" and visualise the results on the screen, or receive them via email (a prompt is shown to provide the email address).

A variant of the standard query, "Run Strict Query" ensures that no non-matching events are found between the events in the sequence. For example, a query can look for a mouse click, a mousewheel event and a keypress, in that order. If run strictly, any sequence of events where a mouse click is found between the mousewheel and the keypress will be discarded. It is important to take into consideration that when the query is translated into a MapReduce query, only the events selected in the sequence are retrieved from the database in order to find possible matches for the query. This filtering significantly reduces the execution time of the query. Once the query is defined it can also be downloaded as an XML file so it can be reused and shared. An example of such file describing a query can be seen in Listing 21.

From XML to Querying the Database To run the resulting query the XML file is first transformed into a MapReduce query that is executed against the Interaction Data Server (see Figure 12). The sequences of events and temporal conditions are converted into JavaScript objects so that the MapReduce algorithm can handle them. In other words, the query sequence is transformed into an array of event objects. For each event, the event type and its context (as defined in the event creation dialogue in Figure 14a) is stored, as many times as its "occurrence" indicates. For example, the sequence described in the Event Sequence Pattern Design area of Figure 13 would be transformed into: [`load`, `mousewheel`, `mousewheel`, `mousedown`/`mouseup`]. The resulting query looks for four matching events, as the value of the occurrence of *Event 2* was 2.

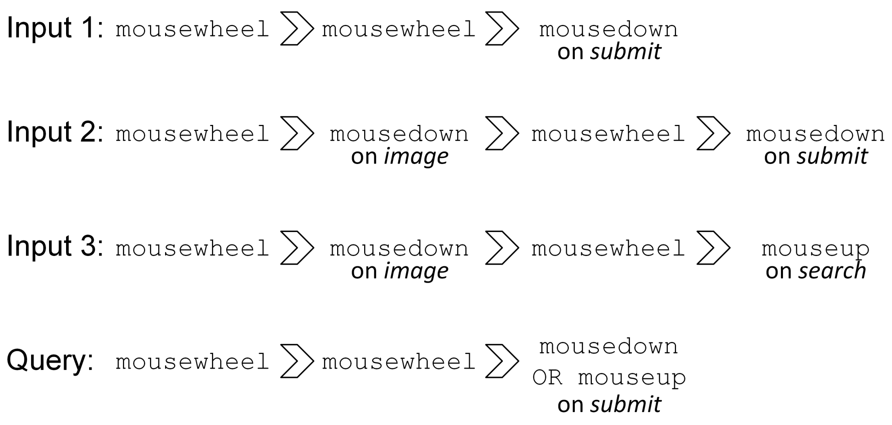

Input 1: mousewheel mousewheel mousedown on submit
Input 2: mousewheel mousedown on image mousewheel mousedown on submit
Input 3: mousewheel mousedown on image mousewheel mouseup on search
Query: mousewheel mousewheel mousedown OR mouseup on submit

Figure 15: How the query matches different inputs through Algorithm 1

The temporal constraints are also transformed, and the indexes of the events affected by the constraint are set to the first occurrence of the corresponding event. The query then searches for the specified pattern of events throughout all users' episodes. WevQuery defines episodes as the interaction from a single user that happen in a Web page without a *noticeable* interruption, where any interruption longer than 40 minutes is considered noticeable. This threshold is consistent with what the literature suggests as far as the definition of episodes is concerned (Jones & Klinkner, 2008; Thomas, 2014). MapReduce is suited for this task as it allows to split the execution of the algorithm for each episode. This way, the algorithm can analyse the interaction events contained in each episode independently.

The algorithm processes every interaction event in the episode in chronological order. Every time the first event in the sequence is matched, a new candidate pattern is created. This candidate pattern contains a queue with the full list of events to be matched. Every time an event is processed, all created candidates check if their next event in the queue can be matched. For an event in the queue to be correctly matched, the processed interaction event must be in the list of accepted event types. If the event contains any context, then this context needs to be matched as well. For example, the list of accepted event types for the Event 3 in the Event Sequence Pattern Design Area in Figure 13 are `mousedown` and `mouseup`, and the `NodeID` field needs to have the value `Submit`. Once all events in a candidate pattern's queue are matched, and as long as the candidate pattern complies with all the temporal constraints, the candidate pattern is stored as a result. If the temporal constraints are not satisfied, then the candidate pattern is discarded. Finally, if the query is "Strict" a mismatch will cause the immediate rejection of the candidate pattern.

Matching Algorithm Example In this example the designer creates a query for the following sequence: `mousewheel` with occurrence value of 2 followed by a `mousedown/mouseup` on a `Submit` button. The query is transformed for the matching algorithm as shown in Figure 15. In the same figure, three different inputs are described, as examples of possible interaction sequences found in an episode. Input 1 represents a successful match, and is accepted as a result at the third step. In Input 2 a successful match is found at the fourth step if the query is not run in strict mode. If the query is strict, the corresponding candidate is rejected at the second step (a `mousewheel` could not be matched). In Input 3, if the query is strict, the corresponding candidate is rejected at the second step. If the query is not strict, the corresponding candidate is rejected at the fourth step as the node value is different to that of the query.

The results of the query, which consist of a JSON and a CSV file with all the matching sequences of events, can be visualised on the screen or sent to the designer via email. Each report contains detailed information about the circumstances in which sequences of events were triggered. The number of episodes in which the pattern is found is reported as well as a list of the details of each matched event. Listing 6 illustrates a report containing one occurrence of a pattern that matched the query. In this simplified report, the context of the occurrence (the user, the URL, and the episode where the occurrence took place) and details of the matched events can be retrieved. Details include temporal information, as well as event specific information. For example, in the case of the page `load` event, the page size is retrieved, and in the case of the `mousedown` event, details about the mouse coordinates, and the target element the user interacted with are provided. In the case of these results, the designer can see that it took the user 1 second to click on the element "Research" after loading the page.

6.2 REST interface

A REST (Fielding & Taylor, 2000) interface has been implemented to provide access to the functionalities of WevQuery. Once a query has been designed using the interface it becomes available in the REST interface so it can be integrated into the ATS.

The interface requires the name of the query to be run as a parameter, and accepts the following optional parameters:

- **userid** If given, the request will only retrieve the results for that particular user.
- **starttime** Indicating a minimum timestamp threshold. If not given, the start of EPOCH timestamp is used.
- **endtime** Indicating a maximum timestamp threshold. If not given, the current timestamp is used, i.e. all events till the moment when the query is executed are included.
- **strictMode** Indicates if the query should be executed in strict mode.
- **fillEventInfo** By default, and in order to optimise the performance of the queries, the only information retrieved for the events in the results are the timestamps, and other information necessary for the execution of the query (such as user id, or the specified context information). After the execution of the query, each event in the results can be filled in with all the information available in the database. This option is false by default, in order to speed up most of the common queries. If additional information for each event is necessary, this option can be set to true.

7 Data acquisition and processing

This section presents the components that have been integrated in the MOVING platform for enabling the collection of data from external sources (that is, in addition to the already ingested MOVING datasets, mentioned in the beginning of Section 4), and the generation of additional or improved metadata for these data by means of data analysis.

7.1 Focused web domain crawler

The Focused web Domain Crawler (FDC) performs the crawling of specific web domains. The users can specify which domains to consider through the Input GUI interface (Figure 16). The domains can also be inserted via command line. However, they are finally stored in the MongoDB database. The FDC checks periodically the database for new inserted domains and launches crawling units called spiders (see details below) that crawl them and extract data from their pages. The webpages' data are later indexed in Elasticsearch in JSON format, as showed in Listing 8, to be searchable in the platform. The FDC creates logs for each of the domains crawled.

The FDC is configured to revisit the domains periodically, filtering the duplicate pages and updating elasticsearch only for the pages with changes in their content. In the old (obsolete) page's document, an "endDate" field is added signifying the date the page stopped existing in it's current form. Thus, documents of pages that haven't been updated don't have an "endDate" field. When the crawler is stopped, it saves the state of its spiders, so when it is restarted it continues the crawling process from where it left.

In terms of implementation, the FDC is based on the Scrapy web crawling/scrapping framework[23]. Scrapy exploits spiders, which are python classes extended to meet the needs of a certain crawling process. Scrapy complies with robots.txt, which is a standard used by websites to inform web bots the areas of the website that can be crawled. We have set minimum crawl delay of 1 second to limit the load on the crawled website. The data collected are indexed using the REST API of Elasticsearch (Listing 7); in this way they are inserted in the MOVING platform. The example in Listing 7 shows the HTTP POST request to index a document. "moving" is the name of the _index and "crawling" the name of the _type for all the crawled data.

So far, the FDC has crawled more than 932.000 pages from a list of 60 websites shown in Table 3.

```
POST http://localhost:9200/moving/crawling
```

Listing 7: Elasticsearch REST API indexing call

[23]https://scrapy.org/, last accessed at 27.09.2017

alison.com	ec.europa.eu	eupartnersearch.com
euraxess.ec.europa.eu	oedb.org	openuped.eu
novoed.com	oli.cmu.edu	oyc.yale.edu
www.open.edu	iversity.org	ocw.mit.edu
www.lynda.com	www.slideshare.net	cloud.imi.europa.eu
mm.fitforhealth.eu	partnersearch.ncps-care.eu	worldmentoringacademy.com
www.net4society.eu	www.ira-sme.net	academicearth.org
ncp-space.net	www.canvas.net	courses.p2pu.org
een.ec.europa.eu	horizon2020projects.com	www.enpicbcmed.eu
www.geoset.info	www.ideal-ist.eu	www.nordplusonline.org
www.transport-ncps.net	welcome.curriki.org	www.edx.org
www.nmp-partnersearch.eu	cordis.europa.eu	janux.ou.edu
lagunita.stanford.edu	open.hpi.de	videolectures.net
www.clustercollaboration.eu	www.coursera.org	www.euresearch.ch
www.innovationplace.eu	www.interregeurope.eu	www.kadenze.com
www.khanacademy.org	www.open2study.com	www.openeducationeuropa.eu
www.openlearning.com	www.pok.polimi.it	www.polhn.org
www.riseba.lv	www.udacity.com	www.up2europe.eu
www.salto-youth.net	www.keep.eu	www.futurelearn.com
www.openculture.com	www.mooc-list.com	www.class-central.com

Table 3: Domains crawled by the FDC.

```
{
"startDate": "2017-08-24",
"fulltext": "<html>...</html>",
"language": "en",
"title": "About | HashiCorp",
"URL": "https://www.hashicorp.com/about/",
"abstract": "HashiCorp is a company based in San Francisco that solves development, operations, and security challenges in
    ↪ infrastructure so organizations can focus on business-critical tasks.",
"docType": "crawled-webpage",
"source": "Web"
}
```

Listing 8: JSON document representing a webpage

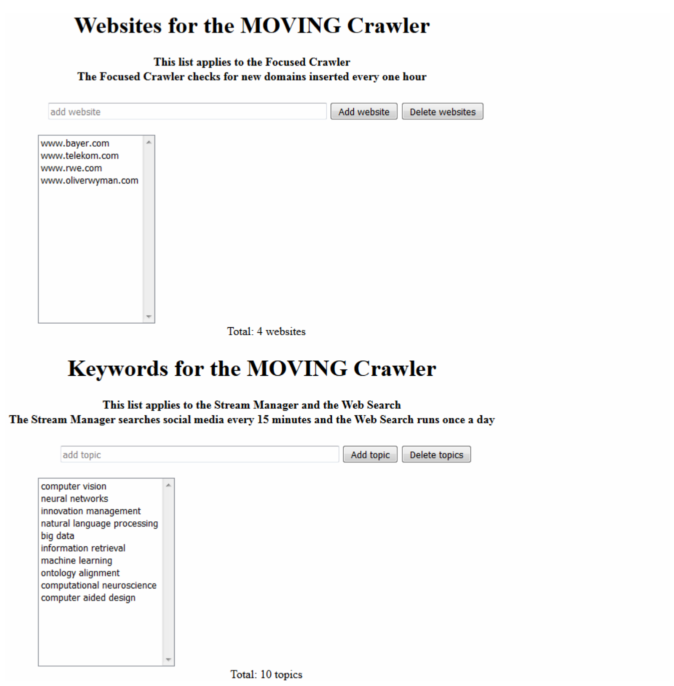
7.2 Search engine-based web crawler

The Search Engine-based web Crawler (SEC) exploits the functionality of web search APIs to collect topic-relevant webpages for the platform. The topics of interest can be specified by the platform users with the use of the Input GUI and are stored in the MondoDB database. Initially, support for Google custom search API and Bing search API was implemented, but later Bing was excluded due to its billing policy. The Google custom search API supports searches in the whole web as well as in specified subsets (i.e. within specific domains). A search engine configuration defines the breadth of the search.

Concerning the SEC's implementation, in order to use the Google API, an API key for authentication must be generated from the Google web console[24] and also a custom search engine configuration has to be created. For the MOVING platform, the whole web is searched. SEC schedules the calls to API (Listing 9) for each topic in rotation, so that all topics are searched until the daily rate limit of 100 API calls is reached. Each API call returns 10 webpage links (Listing 10), as a result there is a maximum limit of 1000 links per day due to the rate limit. For each page retrieved, a JSON document is created and indexed in Elasticsearch the same way as in FDC. The latest implementation of SEC includes log rotation, a process that archives log files to limit their total size while still allowing analysis of recent events. It also encompasses similar duplicate filtering with the FDC.

During its very first testing, the SEC collected just over 1000 webpages for the list of topics shown in Table 4.

[24] `https://console.developers.google.com`, last accessed: 22/09/2017



Websites for the MOVING Crawler

This list applies to the Focused Crawler
The Focused Crawler checks for new domains inserted every one hour

www.bayer.com
www.telekom.com
www.rwe.com
www.oliverwyman.com

Total: 4 websites

Keywords for the MOVING Crawler

This list applies to the Stream Manager and the Web Search
The Stream Manager searches social media every 15 minutes and the Web Search runs once a day

computer vision
neural networks
innovation management
natural language processing
big data
information retrieval
machine learning
ontology alignment
computational neuroscience
computer aided design

Total: 10 topics

Figure 16: Crawler's Input UI.

Information literacy	MOOC	Open innovation
game theory	decision analysis	cryptography
constitutional law	buisiness models	mathematics
geometry	artificial intelligence	deep learning
digital marketing	robotics	chemistry
biology	sociology	climate change
transportations	global warming	

Table 4: Topics crawled by the SSM and SEC.

```
GET https://www.googleapis.com/customsearch/v1?key=KEY&cx=CX&q=car
```

Listing 9: A request to the Google custom search API

```
{
"items": [
{
"title": "Computational geometry -Wikipedia",
"link": "https://en.wikipedia.org/wiki/Computational_geometry"
},
{
"title": "Computational Geometry -Journal -Elsevier",
"link": "https://www.journals.elsevier.com/computational-geometry/"
},
{
"title": "Computational Geometry -ScienceDirect.com",
"link": "http://www.sciencedirect.com/science/journal/09257721"
},
{
"title": "Amazon.com: Computational Geometry: Algorithms and Applications ...",
"link": "https://www.amazon.com/Computational-Geometry-Applications-Mark-Berg/dp/3540779736"
},
{
"title": "Computational Geometry, Algorithms and Applications",
"link": "http://www.cs.uu.nl/geobook/"
},
{
"title": "CMSC 754 Computational Geometry1",
"link": "http://www.cs.umd.edu/~mount/754/Lects/754lects.pdf"
},
{
"title": "Computational Geometry | Mechanical Engineering | MIT ...",
"link": "https://ocw.mit.edu/courses/mechanical-engineering/2-158j-computational-geometry-spring-2003/index.htm"
},
{
"title": "The Computational Geometry Algorithms Library",
"link": "https://www.cgal.org/"
},
{
"title":    "| Computational Geometry | edX",
"link": "https://www.edx.org/course/ji-suan-ji-he-computational-geometry-tsinghuax-70240183x"
},
{
"title": "Computational Geometry",
"link": "http://www.computational-geometry.org/"
}
]
}
```

Listing 10: A reply from the Google custom search API for the topic "Computational Geometry"

7.3 Social stream manager

The Social Stream Manager (SSM) is the crawler responsible for the collection of topic-specific data from social media. The topics are specified from the Input GUI by the platforms users similarly to SEC (Section 7.2). The core component of SSM is the stream-manager, a Java open-source software developed for the FP7 project Social Sensor[25], which is configured to monitor four social media: Twitter, Facebook, Google+ and Youtube. These social media have APIs that the stream-manager exploits to obtain data from them. The stream-manager stores the social media posts it fetches in MongoDB as Items. A Python wrapper starts and stops the stream-manager and fetches URLs contained in the *Items*. It also indexes the pages fetched similarly to the other crawlers.

Log rotation and duplicate filtering have also been implemented for SSM. The latest SSM version also handles social media video URLs. Those are sent to the Video Analysis Service (VIA) for visual analysis (see section 7.6). The analysis results, along with the other video metadata, form the JSON document, is indexed in Elasticsearch (Listing 11). The analysis results for each video are the top visual concepts that are detected in the video by means of their analysis (see Deliverable D3.1 "Technologies for MOVING data processing and visualisation v1.0" Section 3.6).

During its very first testing, the SSM has collected about 400 webpages and 100 videos for the list of topics shown in Table 4.

```
{
"source": "Web",
"docType": "crawled-video",
"URL": "https://www.youtube.com/watch?v=eW3gMGqcZQc",
"title": "What is a MOOC?",
"authors": "dave cormier",
"abstract": "Narrated by Dave Cormier
Video by Neal Gillis -Research by: Bonnie Stewart Alexander McAuley George Siemens Dave Cormier

Created through funding received by the University of Prince Edward Island through the Social Sciences and Humanities
    ↪ Research Council's 'Knowledge Synthesis Grants on the Digital Economy'

CC-BY 2010",
"startDate": "2017-08-30",
"endDate": "2017-08-30",
"concepts": ["Animation_Cartoon", "Commercial_Advertisement", "Graphic", "Synthetic_Images", "Text"]
}
```

Listing 11: A JSON document representing a video

7.4 Bibliographic metadata injection

The Web is a widely accepted source of information for various purposes. The information is primarily presented as text embedded in HTML documents to improve the readability for humans. However, alongside

[25] http://socialsensor.eu/

the commonly known Web of (HTML) documents, there exists an ongoing trend of publishing and interlinking data in machine readable form following the Linked Data principles[26]. Thus, forming the so called Linked Open Data (LOD) cloud[27]. Among other features, LOD allows embedding semantics into search operations. While in the Web of documents we can search for occurrences of the term *book*, in the LOD cloud we can explicitly search for resources which are of type book. However, similarly to classic web search, finding information is a challenging task since there is a vast amount of data available, which is distributed over various data sources. Thus, a search engine with an index of all information present on the LOD cloud is needed. For the bibliographic metadata injection service, we make use a LOD search engine.

Input The service requires as input a SPARQL[28] query using a combination of RDF[29] types and/or properties. SPARQL is the de facto standard query language for Linked Data. This query needs be carefully chosen in order to reliably identify relevant information. In particular, two important aspects need to be considered: the vocabularies and the combination of properties and types from those vocabularies. In the context of LOD, "A vocabulary consists of classes, properties and datatypes that define the meaning of data." (Vandenbussche, Atemezing, Poveda-Villalón, & Vatant, 2017). Thus, choosing properties and types from domain specific vocabularies is crucial to avoid false results. In Listing 12, we present an exemplifying query using the Bibliographic Ontology (bibo)[30] and DCMI Metadata Terms (dcterms)[31], which are two domain-specific vocabularies. However, such vocabularies are often used in an non-conform way, e. g. in a different context (Meusel, Bizer, & Paulheim, 2015). Thus, the combination for specific properties and types from the desired vocabularies is also important. To sufficiently address this issue, we rely on previous analyses of how to explicitly model bilbiographic metadata as LOD (Jett, Nurmikko-Fuller, Cole, Page, & Downie, 2016).

```
SELECT ?x
WHERE {
    ?x rdf:type bibo:Document .
    ?x dcterms:title [].
    ?x dcterms:description [].
    ?x dcterms:creator [].
}
```

Listing 12: SPARQL query using the RDF type bibo:Document and three properties from the DCMI Metadata Terms vocabulary to search for bibliographic metadata.

Furthermore, a mapping file is required. In this mapping file one defines which information is used for which attribute within our common data model (see Deliverable D3.1 "Technologies for MOVING data processing and visualisation v1.0"). The example presented in Listing 13 maps the example SPARQL query to the common data model v1.0. Please note, in this file we do not use the common namespaces (dcterms, bibo), but fully qualified URIs. We can assume that all properties used in the query appear in the datasources we harvest. In addition to these properties, we may specify optional properties in this file, e. g. `http://purl.org/dc/terms/language`. This information can be present in some cases, but does not necessarily have to.

[26] `https://www.w3.org/DesignIssues/LinkedData.html`, last accessed: 18/09/2017
[27] `http://lod-cloud.net/`, last accessed: 18/09/2017
[28] `https://www.w3.org/TR/rdf-sparql-query/`, last accessed: 18/09/2017
[29] `https://www.w3.org/RDF/`, last accessed: 18/09/2017
[30] `http://lov.okfn.org/dataset/lov/vocabs/bibo`, last accessed: 04/09/2017
[31] `http://lov.okfn.org/dataset/lov/vocabs/dcterms`, last accessed: 04/09/2017

```
{"BibItemMapping":{
        "title":["http://purl.org/dc/terms/title"],
        "abstract":["http://purl.org/dc/terms/description"],
        "author":["http://purl.org/dc/terms/creator"],
        "startDate":["http://purl.org/dc/terms/date"],
        "venue":["http://purl.org/dc/terms/isPartOf"],
        "language":["http://purl.org/dc/terms/language"],
        "keyword":["http://purl.org/dc/terms/subject"]
    },
    "AuthItemMapping":{
        "name":["http://www.w3.org/2000/01/rdf-schema#label",
                   "http://xmlns.com/foaf/0.1/name"]
    }
}
```

Listing 13: Example mapping file used to transform bibliographic metadata modelled as Linked Data into MOVING's common data model

Finally, the service requires as input data from the LOD cloud. In the current version, this data is provided as a (crawled) dataset, which is available on a local storage device. Accessing the data directly in the LOD cloud using a LOD crawler is possible and can be implemented in the future.

Functionality This service operates in three steps: (1) query execution, (2) harvesting, and (3) mapping and export. For each step there exists a component responsible for its execution. A high-level view of the components and the data flow is depicted in Figure 17.

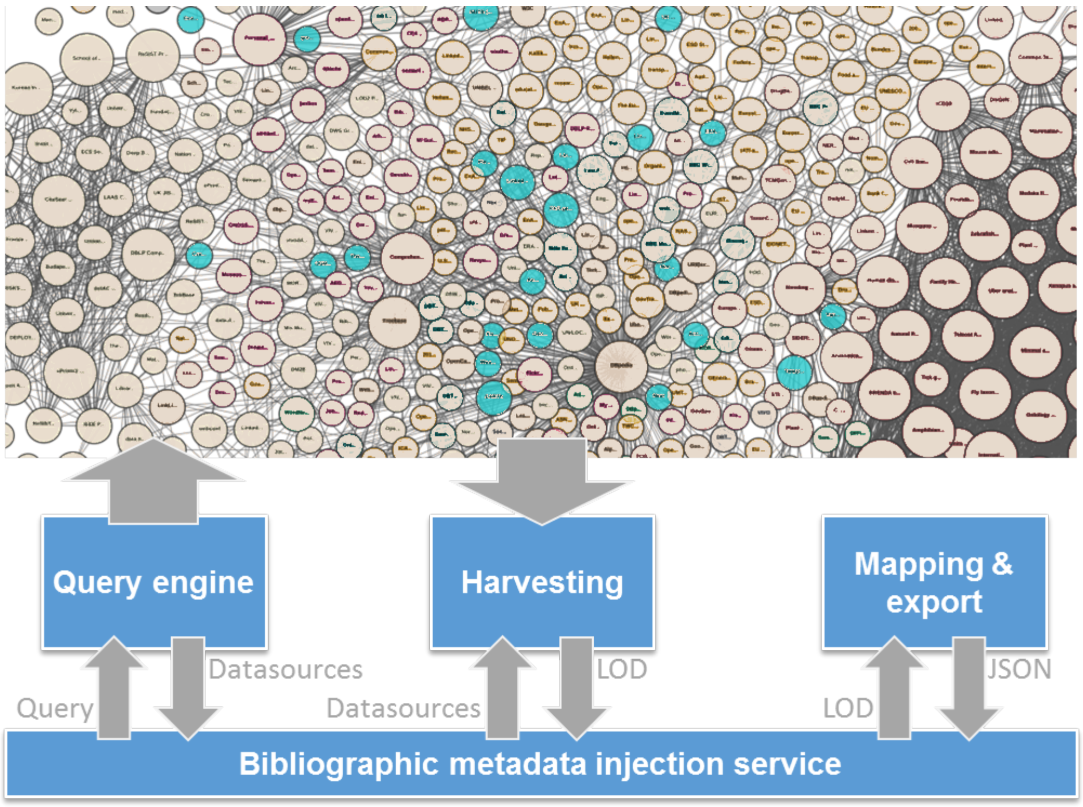


Figure 17: The bibliographic metadata injection service acquiring additional content from the Linked Open Data (LOD) cloud using its core components. The highlighted datasources in the LOD cloud are identified using the query engine, subsequently harvested, and their respective content mapped and exported to JSON objects following the common data model.

As a first step, the query is sent to the LOD search engine LODatio (Gottron, Scherp, Krayer, & Peters, 2013). LODatio supports the execution of queries using a combination of RDF types and properties, since it uses a schema-level index. More details on schema-level indices are available in Section 3.5.4 of Deliverable D3.1 "Technologies for MOVING data processing and visualisation v1.0". The LOD search engine returns a list of identified datasources within the LOD cloud. The LOD cloud can be seen as a distributed knowledge graph, while this list can be considered a sub-graph comparable to a list of HTML websites in classic web search. For instance, in Figure 17, the relevant sub-graph consists of the highlighted nodes. In the subsequent step, the datasources are harvested by means that the contained information is extracted. The information is parsed and converted as specified in the mapping file. Please note, each datasource possibly contains additional information which may be not relevant, e. g. non bibliographic content. However, the mapping file ensures we only parse the desired information. Finally, the converted data is exported into JSON objects following the common data model.

Results The bibliographic metadata injection service is able to retrieve bibliographic metadata modelled as Linked Open Data and transform it into our common data model. Thus, it returns JSON objects, which can be ingested into the Elasticsearch index as additional content. An excerpt of the first generated dataset is presented below. For the initial version of the MOVING platform, we ingested 181,235 bibliographic metadata records harvested from a crawled snapshot of the LOD cloud called Billion Triple Challenge Dataset 2014[32].

```
{
    "identifier":"http:\/\/bnb.data.bl.uk\/id\/resource\/009098350",
    "URL":"http:\/\/bnb.data.bl.uk\/doc\/resource\/009098350?_metadata=all,views,formats,execution,bindings,site",
    "title":"China investment guide :the North-east :establishing and operating a business",
    "authors":[
        {
            "identifier":"http:\/\/bnb.data.bl.uk\/id\/person\/MorarjeeRachel",
            "URL":"http:\/\/bnb.data.bl.uk\/doc\/resource\/009098350?_metadata=all,views,formats,execution,bindings,site",
            "name":"Morarjee, Rachel"
        }
    ],
    "venue":{
        "identifier":"http:\/\/bnb.data.bl.uk\/id\/
            ↪ resource\/009098350\/publicationevent\/HongKongLondonEconomistIntelligenceUnit1997",
        "URL":"http:\/\/bnb.data.bl.uk\/doc\/resource\/009098350?_metadata=all,views,formats,execution,bindings,site",
        "name":"Hong Kong ; London :Economist Intelligence Unit, 1997"
    },
    "source":"BTC2014",
    "docType":"RDF",
    "language":"en",
    "keywords":[
        "Manchuria%28China%29Economicconditions",
    "BusinessenterprisesChinaManchuria",
        "InvestmentsChinaManchuria",
        "Manchuria%28China%29",
        "Manchuria(China)",
        "Manchuria(China)Economicconditions"
    ]
}
```

Listing 14: Example bibliographic metadata record harvested from the Billion Triples Challenge (BTC) 2014 dataset and converted to the common data model.

7.5 Author name disambiguation

In Deliverable D3.1 "Technologies for MOVING data processing and visualisation v1.0" a novel approach to author name disambiguation was presented and explained. This approach was integrated in the context of the MOVING platform by providing a number of python scripts that can be run by an administrator on the platform in order to disambiguate author name mentions on documents stored in the Elasticsearch index. The scripts implementing the current setup are depicted in Figure 18. These scripts are usually applied one after the other in the given order:

1. insert mentionID
2. extract features
3. disambiguate names
4. insert authorIDs
5. "also by this author"

[32]`http://km.aifb.kit.edu/projects/btc-2014/`, last accessed: 18/09/2017

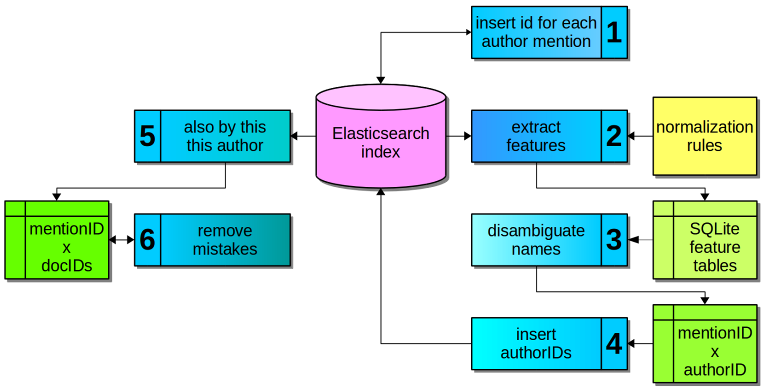


Figure 18: Scripts 1 to 4 disambiguate author names in the Elasticsearch index with the help of intermediate SQLite lookup tables; scripts 5 and 6 enable a feature that suggests to the user documents given a selected author

6. remove obvious mistakes

Scripts (2), (3) and (4) are the most important ones. Script (1) is simply a deterministic process that inserts an identifier (mentionID) to the respective metadata of each author in the Elasticsearch index. We refer to a mention of an author on a document as a *mention*. A mention is unique. Obviously, this is not equivalent to his name. There are persons which are mentioned on different documents, have different names and share the same name with another person. We plan to do this at ingestion time in the future as it is the trivial determinstic procedure of adding the document ID and the number of the author as appearing on the current document. The mentionID gives us a handle for each item that we need to disambiguate. Scripts (2), (3) and (4) make sure that each author metadata also receives an identifier (authorID) that denotes the actual real world person behind the mention – the cluster. Details regarding these three steps follow below. Script (5) precomputes a mapping $mentionID \rightarrow docIDs$. As each mention x in the collection is assigned an authorID, this mention can also be assigned a set of documents (possibly empty) such that each of these documents has a mention x' that has been assigned the same authorID as x during disambiguation.

Script (6) removes obvious mistakes that have happened during the disambiguation process and relate to the problem of *author name synonymy*: two author mentions cannot relate to the same author if neither of the two is a generalization of the other. For example, the name *'John W. Doe'* can not refer to the same person as *'John H. Doe'*. This fact can hardly be taken into acount during clustering as there might be a name *'John Doe'*, which can indeed be clustered with both names. The only alternative is to use a constrained clustering with cannot-link constraints for disambiguation, where even checking if there is a possible solution is NP-hard (Davidson & Ravi, 2005).

In the following, the disambiguation process implemented by scripts (2), (3) and (4) is explained. In Elasticsearch it is computationally and syntactically difficult to look up specific information given another piece of information. For this reason, script (2) creates or updates a look-up table in the form of a single SQLite file which maps directly each mentionID to the respective features retrieved from the Elasticsearch index. Note that in this context, some normalization is applied on most fields (or *feature-types*). Script (3) runs on this database only (no connection to the Elasticsearch index required) and applies the disambiguation code presented in Deliverable D3.1 "Technologies for MOVING data processing and visualisation v1.0". In addition, a list of names can be passed to the script in order to avoid disambiguating all names again. In that case, only the names on the list will be disambiguated. The idea is that this list consists of all names that were added during the most recent data ingestion. The clustering of all other names is independent and does not need to be recomputed. Script (3) adds the authorIDs to the feature database but also creates another SQLite file which maps the mentionIDs of all freshly disambiguated names to the respective authorIDs (clusters). Script (4) inserts these authorIDs into the Elasticsearch metadata records of all author mentions that have a freshly disambiguated name.

In order to make the disambiguation results available to the user, we plan to include a feature into the MOVING platform interaction layer that allows the user to click on the name of an author given for a selected document. This click event then triggers an Elasticsearch query for the authorID assigned to the respective mention – or, if scripts (5) and (6) are exploited, for the set of documents given directly by the precomputed and cleaned $mentionID \rightarrow docIDs$ mapping. Furthermore, in some cases, the set of documents by the selected author can be complemented with existing information from datasets which have been integrated into the platform. This information (which naturally implies annotated authorIDs) could also allow evaluation in terms of Precision and Recall as well as another postprocessing script to correct the thereby identified false positives and false negatives.

7.6 Video analysis

Video Analysis (VIA) is a REST web service that performs temporal segmentation and visual concept detection of videos hosted on social media platforms, such as Twitter, Facebook and Youtube, and file hosting servers as Dropbox. It is possible to communicate with the service via HTTP POST and GET requests, and the processing results can be retrieved in XML and JSON formats. The web service is an external component of the MOVING platform, running on a CERTH server dedicated to MOVING in order to handle the load. The SSM (Section 7.3) sends processing requests to VIA for videos it has crawled and then receives the top visual concepts for the video, which appends to the other metadata that it collects.

Listing 15 shows a processing request to the service. The "shot-scene-concept" URL suffix denotes that the service performs shot and scene temporal analysis, and concept detection on the shot keyframes. The body of the request includes the URL of the video to be processed and a authentication key. The service issues an initial reply upon the delivery of the request, informing about the validity of the request and the identifier (`video_id`) of the video that will be used later (Listing 16).

```
POST http://multimedia2.iti.gr:8080/shot-scene-concept
{
"video_url": <url>,
"user_key": <key>
}
```

Listing 15: Issuing a processing request

```
{
"message": "The REST call has been received. Please check the status of the analysis via the
    appropriate REST call",
"video_id": <video_id>
}
```

Listing 16: Initial service response

To automatically inspect the processing progress of a specific video, a request to its processing status can be issued, adding the "video_id" received from the previous call, as shown in Listing 17. Some of the returned status messages can be seen in Listing 18. The SSM crawler periodically inquires about the status and when notified about the processing having been completed, it issues a new request (Listing 19) to retrieve the results. This REST call returns a JSON file with all the 300 concepts and their confidence scores for the whole video (Listing 20). These scores are produced by performing max pooling to the concept scores of all the video shots. More details on the technologies behind the video analysis component can be found in Deliverable D3.1 "Technologies for MOVING data processing and visualisation v1.0", Section 3.6.

VIA has analyzed 97 videos sent by the SSM with a total duration of 9 hours and 7 minutes.

```
GET http://multimedia2.iti.gr:8080/status/<video\char'_id>
```

Listing 17: REST call to examine the processing status

```
VIDEO_DOWNLOAD_STARTED
VIDEO_DOWNLOAD_FAILED
VIDEO_WAITING_IN_QUEUE
VIDEO_ANALYSIS_COMPLETED
```

Listing 18: A sample of different status messages

```
GET http://multimedia2.iti.gr:8080/concepts_single/<video_id>
```

Listing 19: REST call to retrieve the visual concepts for a video

```
{
"Scene_Text": 0.3112474,
"Helicopters": 0.3839311,
"Horse": 0.2093273,
"Crane_Vehicle": 0.2706041,
"Rescue_Vehicle": 0.2440638,
"Herbivore": 0.2765871,
"Head_And_Shoulder": 0.2670132,
"Mountain": 0.2677047,
"Clearing": 0.2732134,
"Walking_Running": 0.2747194,
...
"Helicopter_Hovering": 0.3671313
}
```

Listing 20: Visual concept scores

8 Conclusion

This report shows how the MOVING platform prototype currently running on a server at the TU Dresden implements what was designed in Deliverable D4.1 "Definition of platform architecture and software development configuration". We have extended the eScience platform of TU Dresden to provide the MOVING functionality. To this end, we added to the MOVING web application a search frontend which includes novel visualisation techniques and a new responsive design.

We have developed an advanced search engine for the MOVING platform. It is based on Elasticsearch which we have augmented with new ranking algorithms like HCF-IDF.

The user training is handled by the Adaptive Training Support. It shows the user visualised statistics about their use of the platform to motivate them users to explore the platform functionalities. It also provides reflection guidance to the users with prompts to facilitate reflection on their search behaviour.

To supply the Adaptive Training Support with data on the use of the platform we have integrated a user interaction tracking system to capture low-level user interaction data. This data is analysed by WevQuery to detect more high-level usage patterns. We have developed a visual designer for the creation of the use patterns to be detected.

We have integrated three content crawlers into the MOVING platform to feed data from specific web sites and social media into the search index. This data is further enhanced by data processing components which can detect concepts in video data, disambiguate authorship information and enrich the search index with additional bibliographic metadata.

In summary, we showed how the first prototype of the MOVING platform, capable of providing an integrated working and training environment for researchers and public administrators, has been implemented. This will soon be followed by the public release of this first version of the MOVING platform, and, in parallel, the development of the final version of the platform, which will be documented in Deliverable D4.3 "Final responsive platform prototype, modules and common communication protocol".

References

Allen, J. (1983). Maintaining knowledge about temporal intervals. *Commun. ACM*, *26*(11), 832–843. Retrieved 2017-01-11, from http://doi.acm.org/10.1145/182.358434 doi: 10.1145/182.358434

Apaolaza, A., & Vigo, M. (2017, June). Wevquery: Testing hypotheses about web interaction patterns. *Proc. ACM Hum.-Comput. Interact.*, *1*(1), 4:1–4:17. Retrieved from http://doi.acm.org/10.1145/3095806 doi: 10.1145/3095806

Davidson, I., & Ravi, S. (2005). Clustering with constraints feasibility issues and the k-means algorithm. In *Proceedings of the 2005 siam international conference on data mining* (pp. 138–149).

Dean, J., & Ghemawat, S. (2008). Mapreduce: Simplified data processing on large clusters. *Commun. ACM*, *51*(1), 107–113. Retrieved from http://doi.acm.org/10.1145/1327452.1327492 doi: 10.1145/1327452.1327492

Fielding, R. T., & Taylor, R. N. (2000). *Architectural styles and the design of network-based software architectures*. University of California, Irvine Doctoral dissertation.

Gottron, T., Scherp, A., Krayer, B., & Peters, A. (2013). LODatio: using a schema-level index to support users infinding relevant sources of linked data. In *Proceedings of the 7th international conference on knowledge capture* (pp. 105–108). Banff, Canada: ACM. doi: 10.1145/2479832.2479841

Jett, J., Nurmikko-Fuller, T., Cole, T. W., Page, K. R., & Downie, J. S. (2016). Enhancing scholarly use of digital libraries: A comparative survey and review of bibliographic metadata ontologies. In *Proc. of the JCDL* (pp. 35–44). ACM. doi: 10.1145/2910896.2910903

Jones, R., & Klinkner, K. (2008). Beyond the session timeout: Automatic hierarchical segmentation of search topics in query logs. In *Proc. of the 17th acm conference on information and knowledge management* (pp. 699–708). Retrieved from http://doi.acm.org/10.1145/1458082.1458176 doi: 10.1145/1458082.1458176

Meusel, R., Bizer, C., & Paulheim, H. (2015). A web-scale study of the adoption and evolution of the schema.org vocabulary over time. In *WIMS* (pp. 15:1–15:11). ACM.

Scherp, A., Franz, T., Saathoff, C., & Staab, S. (2009). F–a model of events based on the foundational ontology dolce+DnS ultralight. In *Proc. of the fifth international conference on knowledge capture* (pp. 137–144). Retrieved 2017-01-12, from http://doi.acm.org/10.1145/1597735.1597760 doi: 10.1145/1597735.1597760

Thomas, P. (2014). Using interaction data to explain difficulty navigating online. *ACM Trans. Web*, *8*(4), 24:1–24:41. Retrieved from http://doi.acm.org/10.1145/2656343 doi: 10.1145/2656343

Vandenbussche, P., Atemezing, G., Poveda-Villalón, M., & Vatant, B. (2017). Linked open vocabularies (LOV): A gateway to reusable semantic vocabularies on the web. *Semantic Web*, *8*(3), 437–452. doi: 10.3233/SW-160213

A WevQuery XML schema

```
<xs:schema  version="1.0.0" attributeFormDefault='unqualified'
     elementFormDefault='qualified'
  xmlns:xs='http://www.w3.org/2001/XMLSchema'>
  <xs:element name='eql'>
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs='unbounded' name='event'>
          <xs:complexType>
            <xs:sequence>
              <xs:element minOccurs='1' maxOccurs='1' name='eventList'>
                <xs:simpleType>
                  <xs:list itemType='eventType'/>
                </xs:simpleType>
              </xs:element>
              <xs:element minOccurs='0' maxOccurs='unbounded' name='context'>
                <xs:complexType>
                  <xs:attribute name='type' type='xs:string' use='required' />
                  <xs:attribute name='value' type='xs:string' use='required' />
                </xs:complexType>
              </xs:element>
            </xs:sequence>
            <xs:attribute name='id' type='xs:string' use='required'/>
            <xs:attribute name='pre' type='xs:string' use='required'/>
            <xs:attribute name='occurrences' type='xs:string' default='1'/>
          </xs:complexType>
        </xs:element>
        <xs:element name='temporalconstraintList' minOccurs='0' maxOccurs='1'>
          <xs:complexType>
            <xs:sequence>
              <xs:element maxOccurs='unbounded' name='temporalconstraint'>
                <xs:complexType>
                  <xs:sequence>
                    <xs:element minOccurs='2' maxOccurs='2' name='eventref'>
                      <xs:complexType>
                        <xs:attribute name='id' type='xs:string' use='required' />
                      </xs:complexType>
                    </xs:element>
                  </xs:sequence>
                  <xs:attribute name='type' type='temporalconstraintType'
                      use='required' />
                  <xs:attribute name='value' type='xs:int' use='required' />
                  <xs:attribute name='unit' type='temporalUnitType' use='required' />
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:simpleType name='eventType'>
    <xs:restriction base='xs:string'>
      <xs:enumeration value='load'/>
      <xs:enumeration value='scroll'/>
      <xs:enumeration value='mouse'/>
      <xs:enumeration value='window'/>
      <xs:enumeration value='mousedown'/>
      <xs:enumeration value='mouseup'/>
      <xs:enumeration value='mouseover'/>
      <xs:enumeration value='mouseout'/>
      <xs:enumeration value='mousemove'/>
      <xs:enumeration value='mousewheel'/>
      <xs:enumeration value='focus'/>
      <xs:enumeration value='blur'/>
      <xs:enumeration value='windowfocus'/>
      <xs:enumeration value='windowblur'/>
      <xs:enumeration value='keydown'/>
      <xs:enumeration value='keyup'/>
      <xs:enumeration value='resize'/>
      <xs:enumeration value='select'/>
      <xs:enumeration value='change'/>
```

```
        <xs:enumeration value='keypress'/>
      </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name='contextType'>
      <xs:restriction base='xs:string'>
        <xs:enumeration value='NodeID'/>
        <xs:enumeration value='NodeType'/>
        <xs:enumeration value='NodeClass'/>
        <xs:enumeration value='NodeTextContent'/>
        <xs:enumeration value='NodeTextValue'/>
        <xs:enumeration value='URL'/>
        <xs:enumeration value='ScrollState'/>
      </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name='temporalconstraintType'>
      <xs:restriction base='xs:string'>
        <xs:enumeration value='within'/>
        <xs:enumeration value='between'/>
      </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name='temporalUnitType'>
      <xs:restriction base='xs:string'>
        <xs:enumeration value='sec'/>
        <xs:enumeration value='min'/>
      </xs:restriction>
    </xs:simpleType>
</xs:schema>
```

Listing 21: XML schema employed by WevQuery