



SATISFACTORY

Project Acronym: **SatisFactory**
Project Full Title: **A collaborative and augmented-enabled ecosystem for increasing satisfaction and working experience in smart factory environments**
Grant Agreement: **636302**
Project Duration: **36 months (01/01/2015 - 31/12/2017)**

DELIVERABLE D3.1

Semantically-enriched framework for analysis and design of dynamically evolving shop floor operations

Deliverable Status: **Final**
File Name: **SatisFactory-D3.1-v1.0-Semantically-enriched framework for analysis and design of dynamically evolving shop floor operations.pdf**
Due Date: **August 2016 (M20)**
Submission Date: **August 2016 (M20)**
Task Leader: **EPFL**

Dissemination level

Public	X
Confidential, only for members of the Consortium (including the Commission Services)	



This project has received funding from the European Union's Horizon 2020 Research and innovation programme under Grant Agreement n°636302

The SatisFactory project consortium is composed of:

CERTH¹	Centre for Research and Technology Hellas	Greece
SIGMA²	Sigma Orionis SA	France
FRAUNHOFER	Fraunhofer-Gesellschaft zur Foerderung der Angewandten Forschung E.V	Germany
COMAU	Comau SPA	Italy
EPFL	École Polytechnique Fédérale de Lausanne	Switzerland
ISMB	Istituto Superiore Mario Boella sulle tecnologie dell'informazione e delle telecomunicazioni	Italy
ABE	Atlantis Engineering AE	Greece
REGOLA	Regola srl	Italy
SUNLIGHT	Systems Sunlight Industrial & Commercial Company of Defensive, Energy, Electronic and Telecommunication Systems S.A.	Greece
GlassUP	GlassUp srl	Italy
QPLAN	Q-PLAN International Advisors LTD	Greece

Disclaimer

This document reflects only the author's views and the European Union is not liable for any use that may be made of the information contained therein.

¹ Project Coordinator

² Terminated beneficiary since June 2016 and replaced by QPLAN

AUTHORS LIST

Leading Author				
#	Surname	First Name	Beneficiary	Contact email
1	Arena	Damiano	EPFL	damiano.arena@epfl.ch
Co-authors				
#	Surname	First Name	Beneficiary	Contact email
1	Jentsch	Marc	FIT	marc.jentsch@fit.fraunhofer.de
2	Ragona	Daniele	REGOLA	d.ragona@regola.it
3	Tsolakis	Apostolos	CERTH	tsolakis@iti.gr
4	Turinetto	Maurizio	REGOLA	m.turinetto@regola.it
5	Vergori	Paolo	ISMB	vergori@ismb.it
6	Voutetakis	Spyros	CERTH	paris@cperi.certh.gr
7	Ziougou	Chrysovalantou	CERTH	czougou@cperi.certh.gr

REVIEWERS LIST

List of Reviewers				
#	Surname	First Name	Beneficiary	Contact email
1	Bougiouklis	Kostas	Q-PLAN INTERNATIONAL	bougiouklis@qplan.gr
2	Krinidis	Stelios	CERTH	krinidis@iti.gr

REVISION CONTROL

Version	Author	Date	Status
0.1	EPFL	October, 2015	Initial Draft
0.2	EPFL	November, 2015	First deployment and test of OSF & ANZO instances (local machines)
0.3	EPFL, CERTH	January, 2016	CERTH Ubuntu Virtual Machine configuration and Ontology Manager deployment
0.4	EPFL	March 2016	Drafting exploitation examples
0.5	EPFL	May, 2016	Overall content review
0.6	EPFL, ALL	June 2016	TOC Review
0.7	EPFL, CERTH	July, 2016	Integration of CERTH/ITI contributions to Section 2
0.9	EPFL	August, 2016	Semantic Context Manager use cases finalization. Final contents check
1.0	EPFL	August 2016	Ready for submission to the EC



TABLE OF CONTENTS

Executive Summary.....	12
1. Introduction	13
2. Deployment Methods and Framework.....	14
2.1 Role of OSF and ANZO.....	14
2.1.1 Data integration	14
2.1.2 OSF stack	15
2.1.2.1 The Content Management Layer - CMS	15
2.1.2.2 The Web Services Layer.....	15
2.1.2.3 The OSF Engines Layer.....	16
2.1.3 OSF OWL API.....	17
2.1.4 OSF Search API.....	18
2.1.5 SPARQL Web Service	20
2.1.6 OSF SW/HW Requirements	22
2.2 Anzo Enterprise.....	23
2.2.1 Anzo Smart Data Platform.....	24
2.2.2 Anzo Smart Data Architecture.....	25
2.2.3 Anzo Smart Data Access	25
2.2.4 Anzo for Excel	27
2.2.5 Anzo SW/HW Requirements	29
2.3 Testing/Deployment Environment	30
2.3.1 ANZO server	30
2.3.2 OSF Server	31
2.3.3 Other	31
3. Semantic Interoperability.....	32
3.1 Alignment with existing ontological resources.....	32
3.2 Context-Driven Information Acquisition	32
3.3 Knowledge Visualization	33
3.4 Knowledge Extraction and Inference.....	35
4. Semantic Enrichment Framework.....	37
4.1 Ontology Manager Architecture.....	37
4.2 Use Cases Analysis	38
4.3 Ontology requirements specifications.....	41
4.4 Integration to SatisFactory platform	41
4.4.1 SatisFactory data flow RDFization	41
4.4.2 Data Exchange Format	43
4.5 IDSS support.....	44
4.5.1 Task duration analysis	47
4.5.2 Worker suitability analysis	50
4.5.3 Experience/Worker Group based analysis	62
4.6 Context-aware engine support.....	65
4.6.1 Analyze historical data to measure the effects of preventive procedures	65
4.6.2 Video tagging to enrich the Context-Aware incident detection videos	68
5. Conclusions	71
References	72



ANNEX.....	73
Annex A.....	73
Annex B.....	76
Annex C.....	78

LIST OF FIGURES

Figure 1: OSF Layered Architecture.....	15
Figure 2 OSF for Drupal Ontologies User Interface based on the OWL API Ontology Library	17
Figure 3 OSF Drupal configuration page	18
Figure 4 Drupal Search API	19
Figure 5 Drupal Search facets.....	19
Figure 6 Satisfactory OSF Virtuoso SPARQL Query Editor	22
Figure 7 Anzo Enterprise supported components	24
Figure 8 Collecting Data for Anzo.....	27
Figure 9 Sharing Data through Anzo	28
Figure 10 Collaborating through Anzo	28
Figure 11 Example of Mock-ups based on context-driven information acquisition mechanisms	33
Figure 12 OntoGraph - OM Overview	34
Figure 13 Mapping concepts: Workers, Worker groups, Skills, and Tasks.....	35
Figure 14 Ontology Manager General Architecture.....	38
Figure 15 COMAU Worker - XML Data RDFzation.....	42
Figure 16 Use of XSL Transformation for CIDEM data RDFization	43
Figure 17 iDSS – OM interaction	44
Figure 18 DSS responds to external Alert.....	45
Figure 19 Black Box: Application examples	47
Figure 20 Black Box: Task Duration analysis.....	48
Figure 21 TDA report - ANZO.....	48
Figure 22 TDA graph – ANZO.....	49
Figure 23 TDA graph - OSF.....	49
Figure 24 Task Duration Analysis example - Task 1.....	50
Figure 25 Black Box: Worker Suitability Analysis	51
Figure 26 Worker Suitability Analysis – Virtuoso SPARQL Query Editor	56
Figure 27 Worker Suitability Analysis - SPARQL query results	56
Figure 28 Worker Suitability Analysis on ANZO	57
Figure 29 Pre-set queries – Sematic Framework	58
Figure 30 OM - SPARQL Query Engine – Semantic Framework	59
Figure 31 WSA results - Semantic Framework	60
Figure 32 WSA graph - Semantic Framework.....	61
Figure 33 MWG Analysis - ANZO	63
Figure 34 SRE Analysis – ANZO.....	63
Figure 35 MWG Analysis – OSF	64



Figure 36 SRE Analysis - OSF.....	64
Figure 37 Context Aware Engine – OM interaction.....	65
Figure 38 ANZO Dashboard example: Filtered Events, Involved Workforce, Measurements	66
Figure 39 Black Box: Historical Data Analysis	67
Figure 40 Context Aware information added to an incident video.....	67
Figure 41 Black Box: Semantics-Enrichment of videos.....	68
Figure 42 - Video enrichment example	70



LIST OF TABLES

Table 1 OSF Minimum Requirements	22
Table 2 ANZO Server Minimum Requirements:	29
Table 3 SatisFactory machine hosting the Anzo Server	30
Table 4 SatisFactory machine hosting the Anzo Server	31
Table 5 Comparison RDF/OWL and XML/XSD	36
Table 6 Analysis of the Use Cases regarding the Ontology Manager.....	39
Table 7 Entities needed for DSS workers classification.....	45
Table 8 List of workers skills.....	52
Table 9 List of skills per worker group.....	53
Table 10 MWG indexes	55
Table 11 SRE Indexes.....	55
Table 12 SRE and MGW analysis fields.....	62
Table 13 Semantic data to measure the effects of preventive procedures.....	66
Table 14 Semantic annotations for incident videos.....	69
Table 15 Detailed description of the worker groups.....	73

LIST OF DEFINITIONS & ABBREVIATIONS

Abbreviation	Definition
AS	Application Scenario
BSC	Business Scenario
CIDEM	Common Information Data Exchange Model
DL	Description Logic
DSL	Domain Specific Language
EC	European Commission
EU	European Union
EXIF	EXchangeable Image file Format
MWG	Matchability index for Worker Group
MIME	Multipurpose Internet Mail Extension
NLP	Natural Language Processing
OM	Ontology Manager
OSF	Open Semantic Framework
OWL	Web Ontology Language
SRE	Suitability index for Requested Experience
UC	Use Case
URI	Uniform Resource Identifier
VM	Virtual Machine



EXECUTIVE SUMMARY

The present document is a deliverable of the SatisFactory project, funded by the European Commission's Directorate-General for Research and Innovation (DG RTD), under its Horizon 2020 Research and Innovation programme (H2020), reporting the results of the activities carried out by WP3. SatisFactory aims to develop an ecosystem of innovative technological components that would assist the daily operations of the people working at industrial environment.

As the outcomes of T3.1, this report describes a **semantically-enriched framework** provided to SatisFactory with the use of **state of the art tools for semantic interoperability**. The underlying architecture will be defined according to the needs of capturing knowledge on dynamically evolving shop floor operations, performed in Task 2.2. Ontologies, for knowledge representation, together with **rules for knowledge extraction and inference**, as well as **graph representation of information** will be some of the tools used to form the necessary framework.

Therefore, this framework will be used to **support user interaction with the Context-aware engine and the Decision Support System**.



1. INTRODUCTION

The SatisFactory approach addresses multi-disciplinary technologies stemming from the deployment of a plug-and-share multi-sensorial framework for collecting effectively tacit knowledge generated in the factory environment, the delivery of a semantically enriched knowledge modelling framework (based on envisioned Common Information Data Exchange Model - CIDEM) for supporting on job education of workers as well as for incident management and proactive maintenance, whereas it provides the capability to automatically verify the correctness of actions performed by the operators in the field. In this context, the deliverable D3.1 describes the semantically-enriched framework for the analysis and design of shop floor operations along with the corresponding software. This framework provides a light weight and full semantic interoperability to support User Interaction with the Context aware engine and the Decision Support System (Task 3.5).

Aiming at achieving context-aware control and re-adaptation of shop floor production facilities for increased productivity and flexibility in use of shop floor resources, a novel multi-sensorial framework is introduced for collecting multi-modal data from the shop floor. The input data streams are aggregated and processed, creating a semantically enhanced base of knowledge, which can be finally utilized to extract intra-factory information concerning production facilities and procedures (machineries, processes, products, production lines, workplaces) and to monitor in real-time the evolving production processes, diagnose problems, flaws and malfunctions (e.g. problems to intermediate or final product). Collected dynamic data are combined with static data describing the production environment, in order to feed the SatisFactory novel model of workplace occupancy evolution and prediction of future needs. The model supports planning and balancing the workload density more efficiently for both the workers and the factory, in terms of balance across available workers in each shift, balance according to production demand highs/lows, etc.

SatisFactory provides an application based on the above framework that is able to decide which information is relevant in a particular situation for a specific user. In order to accomplish that, an ontology-based context model which captures the general concepts about user and business context is developed. A set of rules coming from prior “hands-on” knowledge and previous experience enrich the knowledge model in order to achieve human resource optimization. The use of semantic technologies in T3.1 makes the model both human and machine understandable, addressing the issue based on the dynamically expanding and semantically enhanced knowledge database. Algorithmic tools for object and task recognition supported by augmented reality will then form the basis for employee training on-the-job without the need of the continuous attention of an educator.

Based on the semantic models presented in this deliverable, the DSS will be designed in Task 3.5 in order to make a step forward towards a better understanding of the involved manufacturing processes and operations, the shop floor actors and machinery, the worker’s roles and responsibilities, the maintenance requirements and procedures and the daily production details and flaws. Consequently, requirements for the semantic framework are defined in close collaboration with Task 3.2, Task 3.3 and Task 3.4.



2. DEPLOYMENT METHODS AND FRAMEWORK

In this section the use of the state-of-the-art tools for semantic interoperability will be thoroughly described by taking into consideration their role in the framework of SatisFactory project and, on top of that, particular emphasis will be put on their usability and easiness to access from the end users.

2.1 *ROLE OF OSF AND ANZO*

The SatisFactory Ontology Manager will be integrated into the SatisFactory Framework through an open source software named Open Semantic Framework (OSF), which will enable the deployment of the SatisFactory semantic models and provide semantically enriched results accessible to other components.

To facilitate the integration of the OM with the OSF, thorough testing is performed with an enterprise tool, ANZO, which supports the visualisation of the semantically enriched data as well as an easy-to-configure interface for quick and effective refinements to the ontology models as well as the rules for the OM. Open Semantic Framework (EPFL + CERTH)

The Open Semantic Framework (OSF)³ is an integrated software stack using semantic technologies for knowledge management. It is designed as an integrated framework that combines existing open source software with additional open source components.

The main features of the system can be summarized as follows:

- Data integration across all content and data types
- Semantic data search through and integrated SPARQL engine
- Distributed, differential data access and permissions
- Publishing and managing information
- Remote access through RESTful OSF Web services for importing, updating, deleting, retrieving, etc. information

2.1.1 *Data integration*

Data integration and exposition represents one of the most relevant and effortful feature at the same time. OSF aims to integrate and manage different types of information content and source, such as unstructured documents, semi-structured files, spreadsheets, and structured databases. However, in this framework, the SatisFactory shop floor information and data content is extracted from CIDEM repository, and then, converted according to the canonical RDF data model, enabling common tools and methods for tagging and managing all content. Accordingly, the same applies to any other external content that might be tagged and managed through the Ontology Manager. Ontologies provide the schema and common vocabularies for integrating across diverse datasets. These capabilities can be layered over existing information assets for unprecedented levels of integration and connectivity. All information within OSF may be powerfully searched and faceted, with results datasets available for export in a variety of formats and as linked data.

³ <http://opensemanticframework.org>

2.1.2 OSF stack

The OSF stack consists of multiple layers. In the standard configuration, there is tight integration with Drupal 7 and its leading modules, enabling use of OSF with standard Drupal interfaces and constructs. All interactions with OSF occur via a robust layer of 27 RESTful Web services and their associated APIs (Figure 1), which abstract and simplify how to interact with the stack. The OSF engines layer provides RDF and OWL management capabilities using the proven Virtuoso (RDF), Solr (search), OWL API (ontologies) and GATE (tagging and NLP) standalone applications. Besides Drupal and these engines, all remaining OSF components and Web services have been developed specifically to achieve the complete architecture of the Open Semantic Framework. OSF has been developed over six years and is now in version 3.4

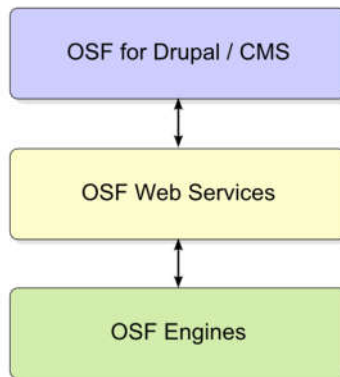


Figure 1: OSF Layered Architecture

2.1.2.1 The Content Management Layer - CMS

Drupal is the standard option packaged with OSF, offering a rich ecosystem of developers and support, plus thousands of modules that extend its functionality and an architecture well-suited to the requirements of OSF. By leveraging existing, well-known Drupal modules and Drupal itself OSF for Drupal manages to expose its full potential to a broad community of developers.

OSF's integration with Drupal occurs via the standard plug-in modules of Drupal and "Drupal connectors" that were designed and built, having as core components conventional Drupal modules, specifically for OSF, extending current, popular Drupal APIs.

Most recently, OSF has also been integrated with simpler, alternative user interfaces such as Bootstrap. This has accompanied the broader use of Clojure in the Web services layer.

2.1.2.2 The Web Services Layer

The OSF stack is controlled or interacted with via its RESTful Web services at the respective layer that can be accessed via dedicated APIs, programmatically, etc.

The OSF Web Services PHP API is a library available to PHP developers to help them generate queries to any OSF Web service endpoint. Each endpoint has its own `WebServiceQuery` class in the API that is used to generate the query, send it to the appropriate endpoint, and get back a resultset. The



resultset can then be manipulated by using the Resultset API. This same API can be used to transform the resultset into different formats. A similar API library is being extended for Clojure.

The `clj-osf` is a simple Clojure Domain Specific Language (DSL) used to query OSF Web Service endpoints. Each of the OSF web service endpoint has its own clojure function. A series of function can be chained to generate a OSF query. That function is used to generate any query, to send it to be endpoint of an OSF Web Service instance and to get back a resultset. The resultset can then be manipulated by using the internal `structEDN` data structure.

These APIs enable developers (or third-party apps such as Drupal) to call functions directly, which then issue the HTTP queries to the respective OSF Web Service endpoints. It is also via the Web Services layer that security and external services may interact with the system. For security, it is possible to either use the native OSF service or invoke an external system.

Because of the central importance of the Web services layer, links to specific services and sandboxes for them are provided under the Web Services menu item.

2.1.2.3 The OSF Engines Layer

The functionality of the Web services layer is based on controlling and interacting with the underlying data engines in the OSF stack. By utilising the common RDF data model, all Web services and actions against the data need to be programmed solely via a single, "canonical" form, thus simplifying the design at the core of the stack and offering a uniform basis to which tools or other work activities can be written. This leads to lower development and maintenance costs, and faster implementation.

The fundamental unit of record aggregation upon which the OSF engines act is the "dataset". A dataset refers to a named grouping of records, best designed as similar in record types and intended access rights. All data objects (e.g. instances, entities, kinds, types or classes), their relations (properties, fields, attributes) and their annotations (metadata) are given Web identifiers in the form of Uniform Resource Identifiers (URIs). This means any and all data within the OSF has a unique identifier, accessible using the HTTP protocol.

The OSF engines are all open source and the respective layer governs the index and management of all OSF content. Documents are indexed by the Solr engine for full-text search, while information about their structural characteristics and metadata are stored in an RDF database, called a "triple store," provided by the Virtuoso engine. The schema aspects of the information (the ontologies) are separately managed and manipulated with their own W3C standard application, the OWL API (see 2.1.3). At ingest time, the system automatically routes and indexes the content into its appropriate stores. Another engine, GATE, is available for semi-automatic assistance in tagging input information and other natural language processing (NLP) tasks.

The OSF engines layer also includes the PHP/Java Bridge, an XML-based network protocol to connect to a Java virtual machine. The bridge gives us the capability to run Java-based engines efficiently within the stack. For efficiency, Web service requests are handled by Memcached. It is an open source, high-performance, distributed memory object caching system. The generic Memcached is an in-memory key-value store for small chunks of arbitrary data (strings, objects), well suited to OSF's API calls.

2.1.3 OSF OWL API

The OWL API [2] is a Java API and reference implementation for creating, manipulating and serialising OWL Ontologies. The latest version of the API is focused towards OWL 2⁴.

The OWL API is open source and is available under either the LGPL or Apache Licenses. The OWL API includes the following components:

- An API for OWL 2 and an efficient in-memory reference implementation
- RDF/XML parser and writer
- OWL/XML parser and writer
- OWL Functional Syntax parser and writer
- Turtle parser and writer
- KRSS parser
- OBO Flat file format parser
- Reasoner interfaces for working with reasoners such as FaCT++, HermiT, Pellet and Racer

For the purposes of the OSF usage, there are also specific function calls within the API for specific ontology component retrievals and manipulations (see Figure 2).

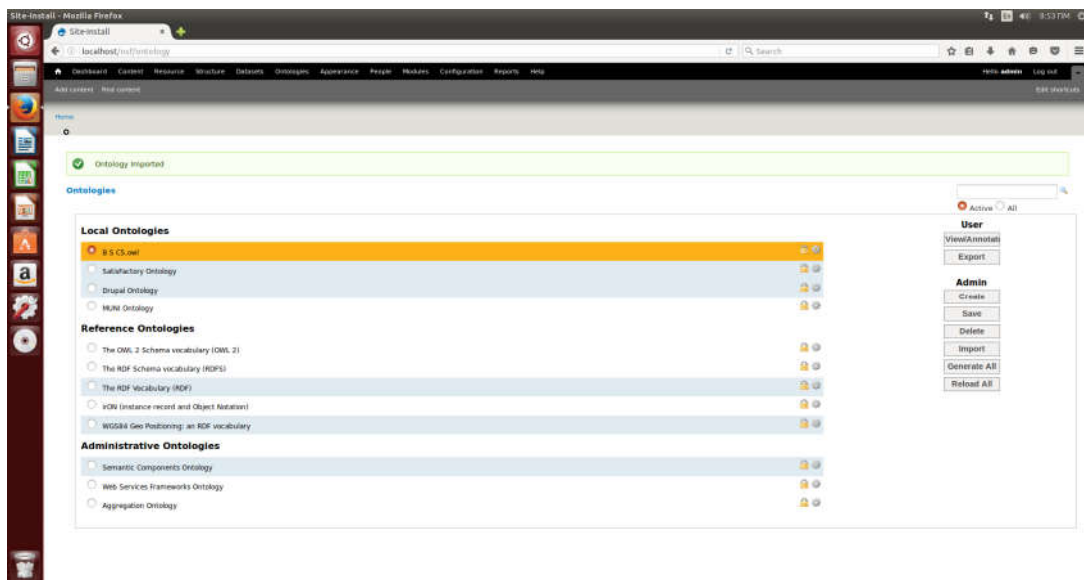


Figure 2 OSF for Drupal Ontologies User Interface based on the OWL API Ontology Library

⁴ <http://www.w3.org/2007/OWL/>

2.1.4 OSF Search API

This section aims to describe how the potentialities of semantic technologies can be leveraged through the use of Drupal. In particular, the main aspects of the semantic framework deployment are thoroughly described in order to provide guidelines and practical examples regarding the configuration of the principal tools provided by Open Semantic Framework for Drupal.

Deployment and Configuration

In general, once you query the system from the main dashboard, then you may get some results that match the researched keyword. A few filtering options can be, therefore, used, i.e. filter by dataset or filter by type, in order to further refine the query outcomes.

Hence, what is crucial from the framework set-up perspective is how the search feature is configured so that it works the way we want it to work and, in particular, it meets the exploitation needs and requirements defined by the end users.

So, to be able to display the proper filtering blocks or to properly reorder results depending on some of their characteristics through the OSF web dashboard, we need to dig down into the Drupal search APIs (see Figure 4) and configure them properly.

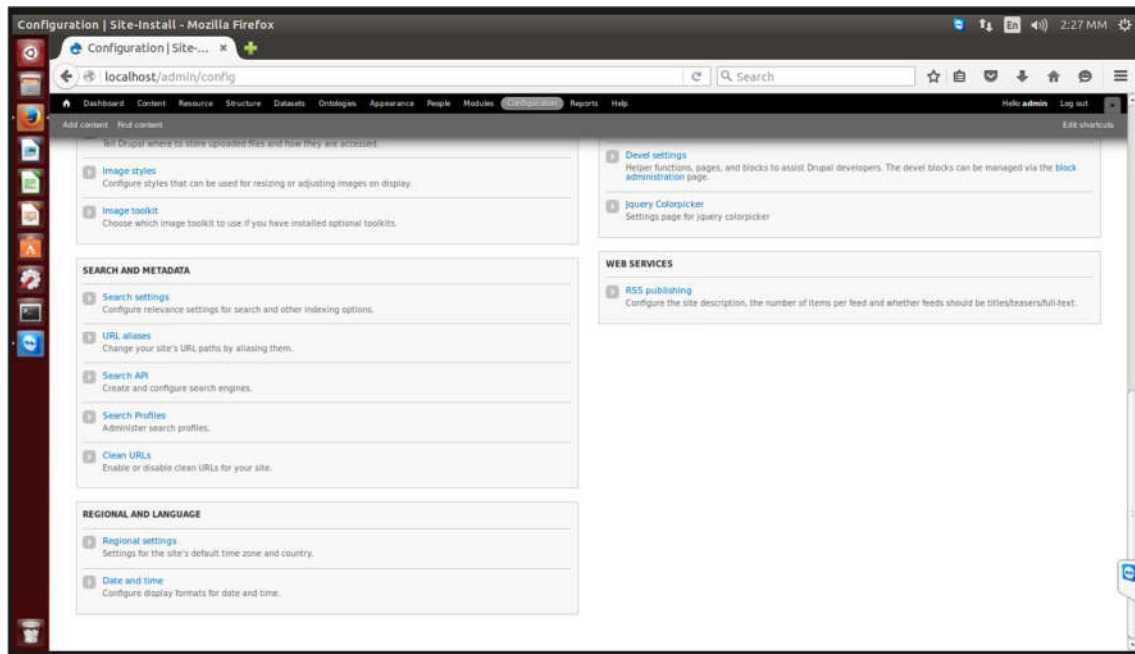


Figure 3 OSF Drupal configuration page

There, it is possible to create multiple Server and Indexes. The Search Server uses a Search Index, which is basically the mechanism used for querying an OSF web service endpoint. The second thing that has to be configured are the search pages, which is basically Drupal page where you go to perform a search query

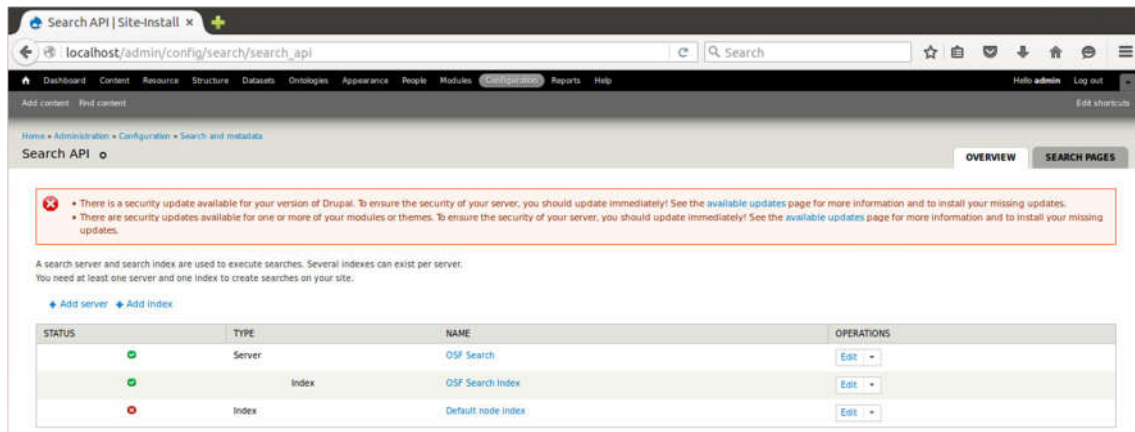


Figure 4 Drupal Search API

Lastly, the Search Index facets have to be configured, in particular, the facets tab allows the developer to define and implement the above-mentioned filtering options that will be displayed in blocks and that will be positioned in any search pages. Every property in OSF can be used as a facet in the search API. These are not displayed in the default configuration tab (see Figure 5) because there is a setting for exposing the available facets into this uses interface. However, there would be too many facets to select there (almost 50 properties have been defined in the OM ontology) and Drupal itself has not been built for that kind of number of facets.

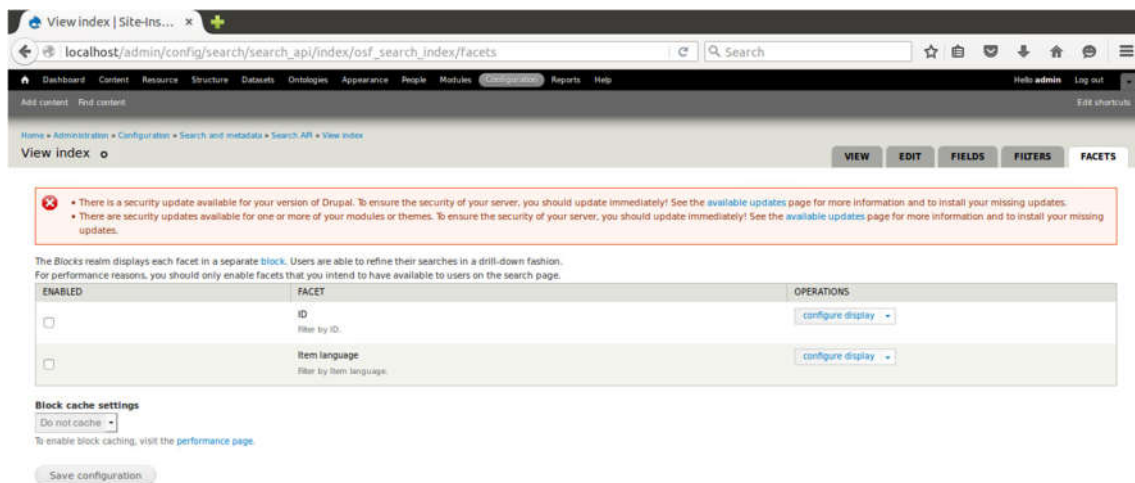


Figure 5 Drupal Search facets

So what should be done before doing that is to go on the “Configure OSF for Drupal modules” section and configure the “Search” tab options. These are settings specific to OSF for Drupal’s search API module. In particular the “Search API Available Facets” tab provides a list of properties being used to describe entities into that system.

2.1.5 SPARQL Web Service⁵

The SPARQL Web service is used to send custom SPARQL queries against the OSF Web Service data structure. This is a general purpose querying Web service.

Developers communicate with the SPARQL Web service using the HTTP POST method which returns one of the following Multipurpose Internet Mail Extensions (MIME) type based on the application:

- text/xml
- application/rdf+xml
- application/rdf+n3
- application/json
- application/sparql-results+xml
- application/sparql-results+json

The content returned by the Web service is serialized using the MIME type requested and the data returned depends on the parameters selected.

If the requested content type sent to the SPARQL web service endpoint is not one of the last two MIMEs (*application/sparql-results+xml* or *application/sparql-results+json*), then it means that the web service is used to return complete record descriptions based on a SPARQL search pattern. Only the two SPARQL result sets serialization formats will return the except SPARQL result sets.

If the SPARQL endpoint is used to get complete records description based on SPARQL patterns, then the three variables (1) *?s*, (2) *?p* and (3) *?o* variables have to be bound in the SPARQL query, otherwise no result will be returned.

Here is an example of a query that match all the records that are *muni:County* and that have an *iron:prefLabel*⁶. However, as you can notice in the query, the three variables (*?s*, *?p* and *?o*) are bound so that the SPARQL endpoint return the complete record description for the requested content type "*application/rdf+xml*":

```
SELECT ?s ?p ?o
WHERE
{
  ?s a <http://purl.org/ontology/muni#County> ;
  <http://purl.org/ontology/iron#prefLabel> ?name ;
  ?p ?o .
}
ORDER BY ?s
```

Optionally, additional bound variables can be defined to get extended results for the triples that defines the records:

```
SELECT ?s ?p ?o ?g (DATATYPE(?o)) AS ?otype (LANG(?o)) AS ?olang ?rei_s ?rei_p ?rei_o
```

⁵ http://wiki.opensemanticframework.org/index.php/SPARQL#Web_Service_Endpoint_Information

⁶ The ontology and the datasets referred in the examples are part of the OSF tutorial and are provided for testing and practising.

```
WHERE
{
  {
    graph ?g
    {
      ?s ?p ?o
    }
  }
  UNION
  {
    graph ?g_rei
    {
      ?rei_s <http://www.w3.org/1999/02/22-rdf-syntax-ns#subject> ?s ;
      ?rei_p ?rei_o .
    }
  }
}
ORDER BY ?s
```

In general, following the SPARQL protocol and a HTTP POST Web service, the OSF engine (through the Virtuoso SPARQL Query Editor – see Figure 6Figure 6) provides a user-friendly and easy-to-use API for querying the OSF datasets. In particular through the use of the following URI:

http://[...]/ws/sparql/ ?query=param1&dataset=&limit=&offset=&default-graph-uri=&named-graph-uri=&interface=&version=

where (all parameters have to be URL-encoded):

- **query** - The SPARQL query to send to the Web service endpoint
- **dataset** - URI referring to a target dataset to query.
- **limit** - Limit of the number of results to return in the resultset (*max 2000*)
- **offset** - Offset of the "sub-resultset" from the total resultset of the query
- **default-graph-uri** - Dataset to target with the sparql query (optional) -- only used for consistency with the SPARQL protocol
- **named-graph-uri** - Dataset to target with the sparql query (optional) -- only used for consistency with the SPARQL protocol
- **interface** - Source interface used for this web service query. The interface is a different way to process a query (different algorithms, different data management system, etc. The default interface is 'default')
- **version** - Version of the interface to query

When a query is sent to this SPARQL endpoint, the user has to specify the dataset he wants to query, and he has to have access to it. If the dataset URI is not specified, or if he doesn't have access to it, then an error will be reported by the endpoint.

To specify the dataset he wants to send the query against, the user can specify it via the endpoint parameters dataset, default-graph-uri or named-graph-uri. Also, he can specify it directly within the SPARQL query via the clauses FROM and FROM NAMED.

Finally, to access the OSF WSF (Web Service Framework) the user needs to have specific permissions in order to avoid unattended access to sensitive information. Therefore, to access the Web service endpoint the user needs proper CRUD (*Create, Read, Update and Delete*) permissions on a specific graph (dataset) of the WSF.

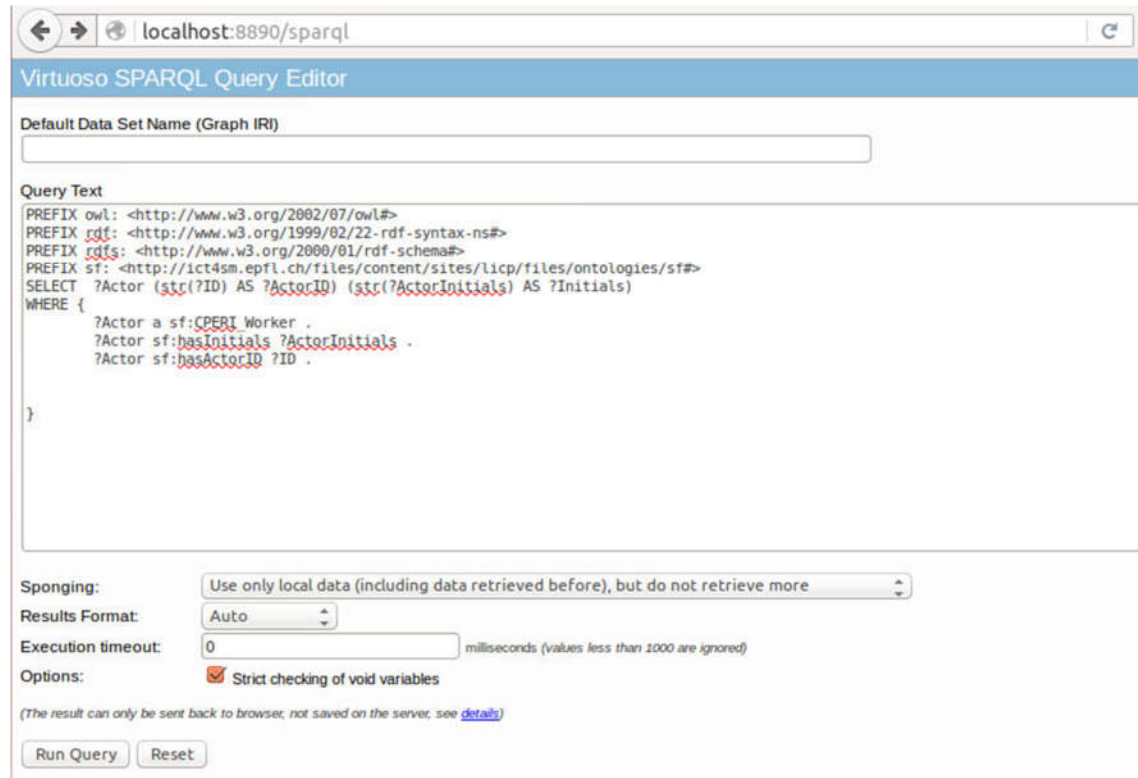


Figure 6 SatisFactory OSF Virtuoso SPARQL Query Editor

2.1.6 OSF SW/HW Requirements

In order to install, configure and access the OSF the following SW/HW requirements are needed:

Table 1 OSF Minimum Requirements

CPU	No restrictions
Architecture	64-bit*
Operating System	CentOS 7 CentOS 6 Ubuntu 14.04
PHP version	PHP 5.6 or higher
Drush version	Drush 8.0 or higher



Virtual vs. physical hardware	Either
Available RAM	2 GB
Disk space	5 GB (<i>Partition where you are installing OSF</i>)
Disk type	No restrictions
Internet access	Yes
Web browser & version	No restrictions

For the Satisfactory project a series of Virtual Machines (VM) with OS: Ubuntu 14.04 were used towards installing, configuring, testing and validating the OSF installation.

2.2 ANZO ENTERPRISE

Anzo Enterprise [1] is an enterprise class software based on Semantic Web Technologies for Smart Data management and advanced Smart Data analytics.

The Anzo Enterprise software can be used for data integration, search, analysis, visualization, and interaction. The collection of Anzo modules is also well-suited to building agile, real-time applications that integrate with varied data sources, and allow for easy customization and evolution as business environments change providing significant end-user self-service.

The Anzo® Smart Data Platform is an open platform for building Smart Data Analytics™ solutions driven by semantic graph technologies. Smart Data Analytics solutions let users link, integrate, discover, search, analyse, visualize, and curate data from any structured or unstructured source internal or external.

Anzo combines the flexibility of an underlying graph database with tools that make it easy for anyone to add new data on the fly with powerful user-driven text, graph and big data analytics. Anzo Smart Data Analytics solutions can be tailored to specific business purposes, focus on end-user self-service with any data from anywhere, can be built and evolved quickly and flexibly and handle data at big data scale. Anzo Smart Data solutions allow companies to easily add third-party analytics, machine learning, graph analysis and other tools to provide additional analytics value. Supported components are shown in Figure 7.

SUPPORTED COMPONENTS	ANZO@EXPRESS	ANZO@ENTERPRISE
Anzo@Smart Data Server	✓	✓
Anzo@for Microsoft Excel	✓	✓
Anzo@on the Web	✓	✓
Anzo@Client Interface	✓	✓
SPARQL Endpoint	✓	✓
LDAP/Active Directory Support	✓	✓
Anzo@Ontology Editor	✓	✓
Anzo@Network Visualizer	✓	✓
Anzo@Formulas	✓	✓
<u>Anzo@Unstructured</u>	<i>Trial</i>	<i>Add-On</i>
Anzo@Connect	-	✓
Anzo@Relational Replica	-	✓
Anzo@Salesforce Connector	-	✓
API Access	-	✓
Federated Directory Support	-	✓

Figure 7 Anzo Enterprise supported components

Anzo Enterprise Features

- Easy Data Aggregation & Management from Internal & External Systems
- Customizable Dashboards
- Semantic Search
- Entity Linking, Investigation & Management
- Integrated Rules, Workflows & Alerts
- Unstructured Text Mining & Analytics: Documents, Email, Web, Social Media
- Enterprise IT Capabilities: Data Lineage, Security

2.2.1 Anzo Smart Data Platform

The Anzo Smart Data Platform [3] is a semantics standards based set of software and tools combined with a mod-ern, enterprise-class services oriented and event driven architecture. This platform sets the foundation for building and deploying Smart Data solutions. These solutions are tailored to specific business purposes, focus on end-user self-service, deal with any data from anywhere and can be built and evolved quickly and flexibly. Anzo Smart Data solutions are made possible by turning data into Smart Data, by leveraging industry-defined semantic models or company-defined models to link and manage diverse data. The semantic models are graph models. They allow for the data to be linked by business concepts and allow for the metadata to travel with the data.



The Anzo Smart Data Platform was built for easy integration and use of third party software modules to provide additional value-add capabilities. Cambridge Semantics has already integrated software modules from a variety of vendors and partners to allow for improved crawling, analytics, text processing, machine learning and language translation. Enterprises can use these modules to improve their analytics and data processing.

2.2.2 Anzo Smart Data Architecture

The Anzo Smart Data Server lies at the heart of the Anzo architecture [4]. The server follows a distributed Service Oriented Architecture (SOA) pattern in which all system components are connected together by a secure asynchronous messaging fabric. This approach has many advantages, foremost of which is the ability to scale up the server across a hardware cluster. The component services can then be executed in parallel for greater throughput. Another significant advantage of this design is the ease with which new Anzo services may be added, including integration with third-party products, legacy systems and remote Web services.

The Anzo Smart Data Server supports the W3C's semantic standards (RDF, RDFS, OWL, SPARQL) at every level of the software stack. This makes Anzo a unique platform for the native support of sophisticated, next-generation, semantically-enabled applications.

The Anzo Smart Data Server provides a (RDF) graph database and a binary content repository. On top of these components, Anzo provides various registries and service directories for client-side applications and system use. For example, Anzo includes an ontology registry for semantic content models and a Web service directory to facilitate discovery and invocation of services.

User and role management can integrate with an organization's existing Active Directory Server or LDAP directory or can be handled via the Anzo Server's included LDAP directory server.

Anzo includes an extensible architecture on both the server-side and the client-side. On the server, business logic may be developed in a number of programming languages including Java, JavaScript and PHP. A client-side API is available for .NET, Java and JavaScript (in the Web browser).

Anzo is architected to ensure that applications written for the server are as scalable as possible. In particular, the architecture implements data replication and multiple levels of disk, memory and client-based caching. Data is transparently replicated to client-side databases in order to make local API access as fast as possible and avoid expensive round trips to the data server. All replicated data is automatically synchronized in near real-time with the server. Up-date transactions use an optimistic-concurrency approach that allows scaling in parallel.

The Anzo Smart Data Server is a Java server with component services that execute in one or more OSGi runtimes connected via the Java Messaging Standard (JMS), allowing for cross-platform deployments. The server includes an administrative console and supports the standard JMX management interfaces and a J2EE compatible servlet container. Anzo Server

2.2.3 Anzo Smart Data Access

Smart Data Access [5] is a new paradigm in using data to make rapid, informed, high-quality decisions across an enterprise. Smart Data solutions are marked by these characteristics:

- Deal with any data, including non-conventional data
- Focus on end-user self-service



- Low cost to build
- Evolve quickly and flexibly
- Tailored to specific business purposes

Let's look at each part of the phrase in turn:

A key value of Smart Data solutions is the unified view of information that they provide. There are three increasingly valuable ways in which a Smart Data solution can unify information:

- **Aggregated.** Aggregated information is information that's been brought together into one place. Instead of asking users to log in to a dozen of different databases and systems and run a dozen of different searches to find everything they want to know about a topic, a UIA solution will provide a single, complete view of all information known about the topic. This saves end users a great deal of time and reduces the chance that a key piece of information goes unnoticed.
- **Harmonized.** Harmonized information has been merged on common entities and concepts. It's not enough to aggregate information, if we're unable to tell that IBM and International Business Machines are the same company. A unified approach to Smart Data access means that all occurrences of any particular topic are harmonized, regardless of how they're identified in different sources.
- **Integrated.** Integrated information is directly linked to related information to facilitate powerful search and analytics. Integrated information allows users to easily perform complex searches and calculations based on attributes of a topic that come from different sources.

Enterprise data integration software has been integrating information for decades, but conventional ETL, EII, and EAI technology are severely limited in the types of information that they can work with. A hallmark of Smart Data Access, on the other hand, is the breadth of information that can be easily accessed:

- **Structured data.** This is the fixed, predictable and well-understood data that conventional data integration usually deals in: data from relational databases, from Web services, from proprietary APIs, etc.
- **Unstructured data.** Unstructured data is data pulled from text content. This is information buried inside of documents, presentations, emails, Web pages, social media sites, etc.
- **Semi-structured data.** Semi-structured data occupies the nebulous area in between structured and unstructured data. Spreadsheets are a prime and ubiquitous example of semi-structured data, as they regularly mix structured tables of data with critical data sprinkled in arbitrary cells throughout a worksheet. Other examples of semi-structured data include mixed XML content and tables of information embedded inside web pages.

A unified approach to information from any source is helpful, but unless that information can be accessed as needed to solve the particular business problem at hand on any given day, its value is limited. Smart Data Access is the key to making unified information as valuable as it can be, and it includes:

- **Any location.** Increasingly, more and more of the data that end users need is owned by someone else. Key pieces of data might reside in another business unit's database; or they may be in a spreadsheet on a colleague's desktop; or in a supply-chain partner's ERP system; or the data might be part of a subscription data-base from a content vendor; or perhaps the



data is publicly available on the web. UIA solutions provide access to data regardless of where it is.

- **Anyone.** Smart Data access is not just for IT personnel; instead, Smart Data solutions allow non-technical end users—such as business analysts, executives, LOB directors, etc.—to pull together information on their own
- **Anytime.** Users rarely know in advance what data they'll need or how they'll need to use it. With Smart Data Access, users see rapid turnaround times, bringing in new sources of information in minutes or hours rather than weeks or months.
- **Any access path.** Smart Data Access doesn't only refer to how the data is originally retrieved. It also refers to giving users a myriad of ways to consume the unified information, ranging from monthly reports to BI dash-boards to web search to Excel spreadsheets, to SQL or MS Access and much more.

2.2.4 Anzo for Excel

Critical information is trapped in Excel spreadsheets on shared drives, in SharePoint, and scattered across laptops. Related data uses different headers, layouts, and positions across different spreadsheets. With Anzo, it is possible to link together data across spreadsheets regardless of what the spreadsheets look like or where they are. Best of all, if a change is made to a piece of data, that data is updated everywhere it occurs—in spreadsheets, databases, and dashboards.

Anzo for Excel is a plugin for existing Microsoft Excel 2003, 2007, 2010, or 2013 installations. This plugin enables many different users to select, collect, combine, share, and reuse the data contained in various spreadsheets on their individual computers. In particular Anzo for Excel offers:

- Data harmonisation across hundreds of disconnected spreadsheets
- Data integration regardless of headers, layout, or where the spreadsheet resides
- Up-to-date spreadsheets in real-time
- Complete picture of relevant information by augmenting Excel data with other sources of enterprise data

For example, users can collect and report on quarterly sales forecasts from geographically dispersed sales teams (Figure 8).

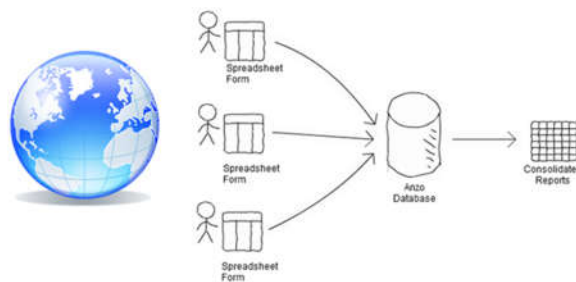


Figure 8 Collecting Data for Anzo

Users can share consolidated workbook data on the web. For example, knowledge workers can publish Excel-based research results on the web for analysts to scrutinize (Figure 9).

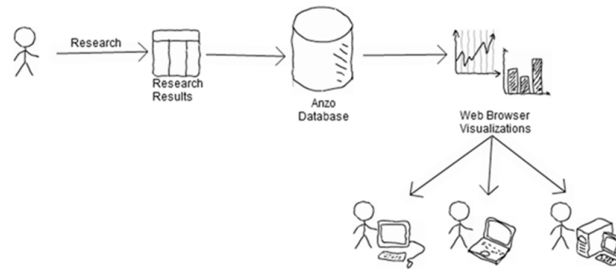


Figure 9 Sharing Data through Anzo

Users can collaborate on workbook data. For example, project team members can work concurrently on workbooks tracking the project's status and deliverables (Figure 10).

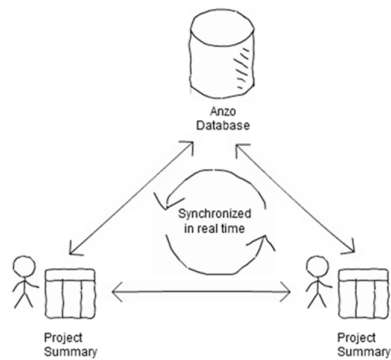


Figure 10 Collaborating through Anzo

2.2.5 Anzo SW/HW Requirements

In order to install, configure and access Anzo Enterprise the following SW/HW requirements are needed:

Table 2 ANZO Server Minimum Requirements:

CPU	Dual-core
Architecture	64-bit*
Operating System	RHEL 5 Windows 2003/2008
Virtual vs. physical hardware	Either
Available RAM	6 GB
Disk space	10 GB (Anzo) 100 GB (Data) (Server)
Disk type	SATA
MS Excel version (Anzo for excel)	Excel 2003
Internet access	Yes
Web browser & version	Internet Explorer 9 Firefox 3 Chrome 2 Safari 4

For the SatisFactory project an Ubuntu VM was used at EPFL premises (intranet) towards hosting the Anzo Enterprise which was used for creating, testing and visualising the SatisFactory Ontology Manager prior to the open source OSF installation.

2.3 TESTING/DEPLOYMENT ENVIRONMENT

2.3.1 ANZO server

Due to licensing restrictions, Anzo Server has been deployed, and then works only, through the EPFL intranet.

Table 3 Satisfactory machine hosting the Anzo Server

CPU	Dual-core
Architecture	64-bit*
Operating System	RHEL 5 Windows 2003/2008
Available RAM	6 GB
Disk space	10 GB (Anzo Server) 100 GB (Data)
Disk type	SATA
MS Excel version (Anzo for excel)	Excel 2003
Internet access	Yes
Web browser & version	Internet Explorer 9 Firefox 3 Chrome 2 Safari 4
Access	Login restrictions EPLF intranet

2.3.2 OSF Server

Located at CERTH premises, the OSF Server is an Ubuntu VM hosted in a PC at CERTH/CPERI premises. The OSF Server has the following specifications:

Table 4 SatisFactory machine hosting the Anzo Server

CPU	Dual-core
Architecture	64-bit
Operating System	Ubuntu 14.04
PHP version	PHP 5.6
Drush version	Drush 8.0
Disk type	SATA
Internet access	Yes
Web browser & version	Firefox Chrome
Access	Team Viewer 10

2.3.3 Other

In order to finalise the installation of the Open Semantic Framework, various installations at multiple VMs took place. Given the irregularities that were observed during the OSF installation a second PC with the same characteristics was set up as backup to the OSF Server hosted at CERTH/CPERI premises.



3. SEMANTIC INTEROPERABILITY

Besides being a cornerstone for defining a knowledge domain, the Satisfactory Network of Ontologies, more specifically the component called “Ontology Manager”, will interact either directly or indirectly with several SatisFactory components. For example, data gathered from many different sources will be stored in the SatisFactory repository according to CIDEM format, and then, enriched with semantics according to the ontology models. This, on the one hand, allows to achieve semantic interoperability and integration, which consequently enables the support user interaction with the Context-Aware Engine and the Integrated Decision Support System.

In the following section, we outline how the Ontology Manager will be exploited either in general terms or specific interaction with the other SatisFactory components.

3.1 ALIGNMENT WITH EXISTING ONTOLOGICAL RESOURCES

A key benefit of semantic technologies is the possibility to adopt and extend existing ontological resources and meta-data initiatives being standards-based, bridging thus multiple domains specific knowledge: environmental, mechatronic, etc. Ontology alignment consists of the identification if the concepts belonging to the to-be-matched pairs of ontologies are related to each other via a *sub-concept* or an *equivalence* relationship.

An initial work in terms of manual alignment has been conducted in collaboration with CETH/ITI, ATLANTIS and ISMB, as part of T2.2, T3.3 and T3.5, in order to align the **SatisFactory Ontology** with specific industrial domains, taking inspiration from the approach used in several research works where such alignment is achieved with an existing ontology modeling the domain of features-based product design [6] [7]. Therefore, as it will be explained in Section 4 (and further documented in D2.1), a wide ontology network (we can refer it as an Upper Level Ontology) may be obtained by linking, and then reusing, existing ontologies in order to enrich the whole semantic structure. Further details will be given in the next section.

3.2 CONTEXT-DRIVEN INFORMATION ACQUISITION

Context is defined as any information that can be used to characterize the situation of an entity [8]. In the context of SatisFactory, the ontology will enable retrieving contextual relationships behind an entity. An entity is a person, place, or object that is considered relevant to the interaction between the user and an application. Thus, SatisFactory sensors should provide sets of shop floor information data that can be exploited in order to extract context-driven knowledge through the application of rules, and then derive the relevancy of those elements in specific situations.

Figure 11 illustrates an example of Shop floor and Management views. This example has been extracted from LinkedDesign [9] and gives a practical idea of context-driven info exploitation. From a management perspective, information such as part costs or a critical disturbance status of the part is relevant. Whereas, from a shop floor perspectives, different information related to the same part can be derived such as failure rate or critical failure status.

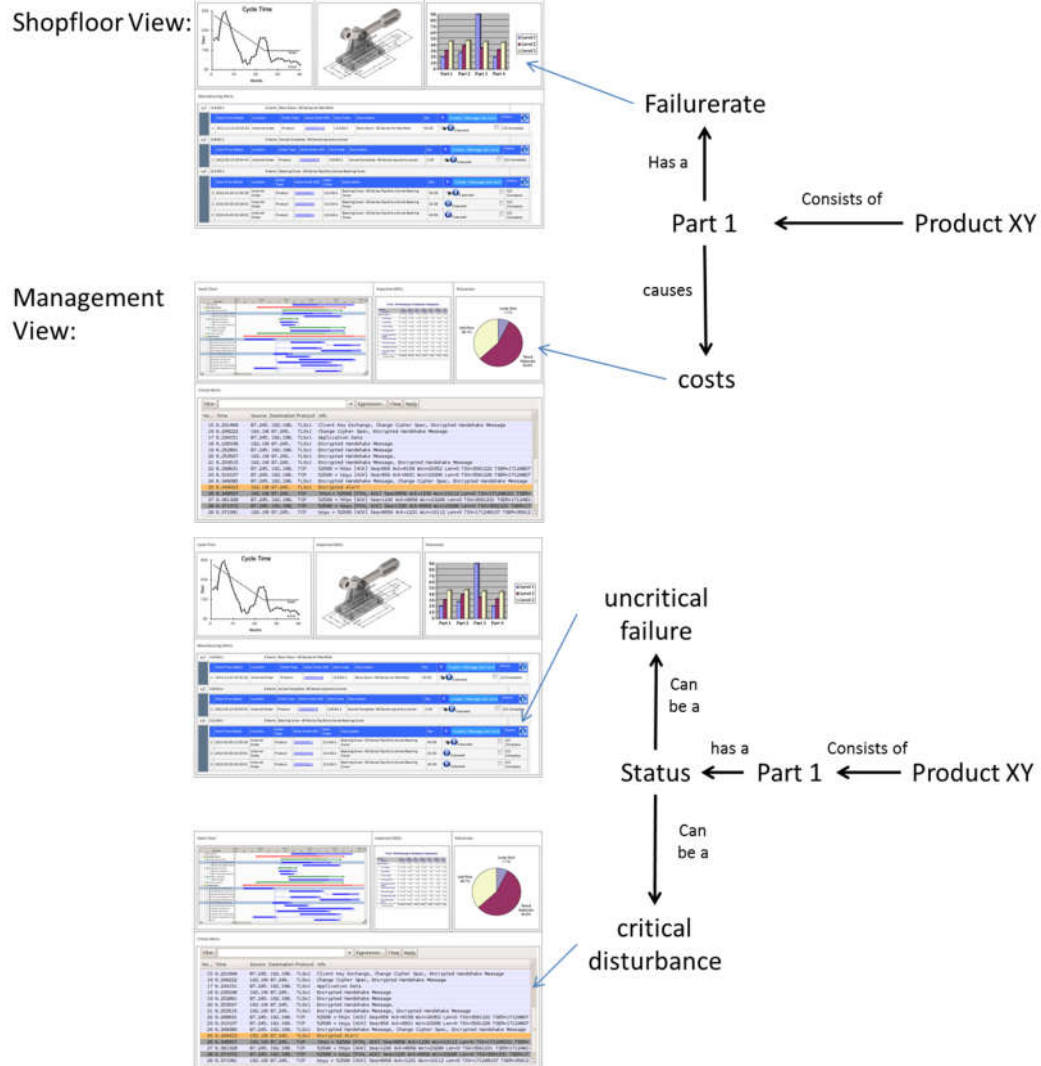


Figure 11 Example of Mock-ups based on context-driven information acquisition mechanisms

3.3 KNOWLEDGE VISUALIZATION

The Ontology captures knowledge that exists in the domain. This implicit and explicit knowledge is of great value for people who are trying to understand the domain. By visualizing the ontology, this knowledge may become much easier to understand. The knowledge can be visualized as a graph. The nodes of this graph can show the different concepts in the domain of interest and edges can demonstrate the various relations between concepts. In this way, the several interested parties can navigate through domain knowledge and come into conclusion regarding the nature of their problem (or simple question) more easily.

Some tools have already been developed towards this direction, but they lack user friendliness and expressivity. Well known examples are OntoViz and OntoGraph (see Figure 12), plugins for Protégé ontology editor.

More advanced cutting edge solutions focus more on the visualization tools. Such tools become more powerful by including more expressivity to existing graph representation techniques. A big challenge in this field is how to include more knowledge elements in one single frame and at the same time keep the picture simple enough as to be understood. An example of the before mentioned tools is included in the OSF. The OSF brings together several semantic technologies to create a software stack where the Ontology plays the role of the brain.

D3.1 – Semantically-enriched framework for analysis and design of dynamically evolving shop floor operations ■ August 2016 ■ SatisFactory project ■ GA #636302

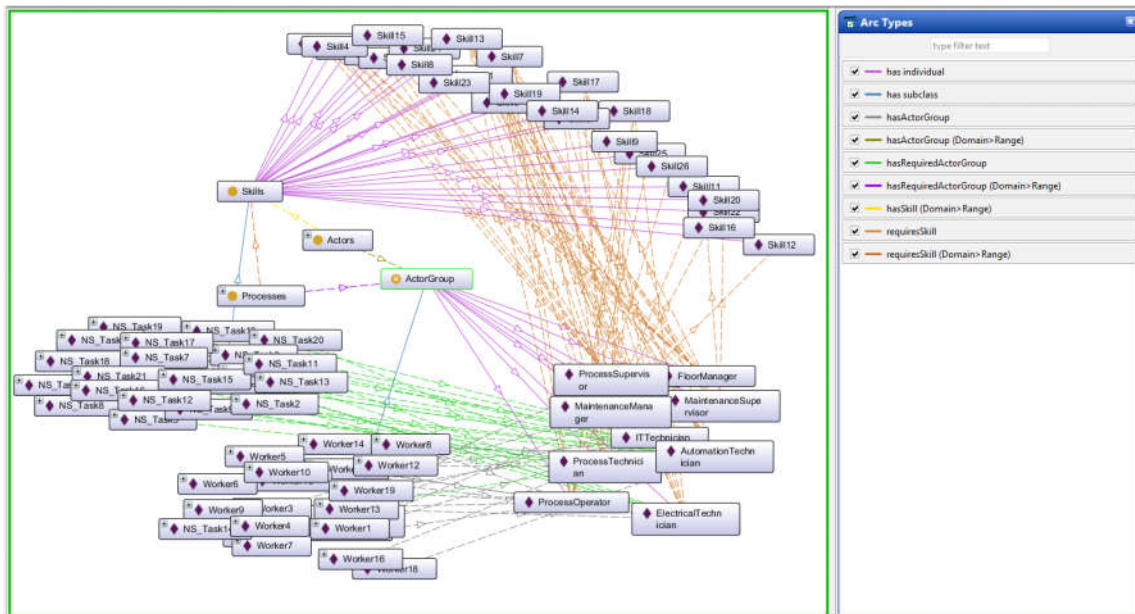


Figure 13 Mapping concepts: Workers, Worker groups, Skills, and Tasks

3.4 KNOWLEDGE EXTRACTION AND INFERENCE

Knowledge Extraction means the creation of knowledge from structured (relational databases, XML) and unstructured (text, documents, images) sources. The resulting knowledge needs to be both machine-readable and machine-interpretable, therefore, it must represent knowledge in a way that unambiguously defines its meaning and facilitates inferencing. Although, it is methodically similar to information extraction (NLP) and ETL (Data Warehouse), the main distinguishing criteria is that the extraction result goes beyond the creation of structured information or the transformation into a relational schema. It requires either the reuse of existing formal knowledge (reusing identifiers or ontologies) or the generation of a schema based on the source data. Schema (or Ontology) generation from legacy sources is a weaker form of knowledge extraction when compared to reusing identifiers. Although hierarchies or OWL/DL axioms of varying expressivity are generated no disambiguation is created by linking the newly created classes to existing ontologies [9].

Therefore, the application of the Linked Data principles for data integration, created and maintained in order to explore the shop floor knowledge, appears to be an added value for Satisfactory. One of the key benefits of Semantic Web technologies, as key enablers of Linked Data, is the creation of data stores using URIs for identifying resources and their relations. A thorough list of characteristics related to both XML and OWL is following presented. Table5 shows a comparison between such modelling languages based on 10 key points [11].

Table 5 Comparison RDF/OWL and XML/XSD

Characteristic	RDF/OWL	XML/XSD
Data structure & model	Statements represented as subject-predicate-object triples, using IRIs, blank-nodes and literals as components.	Tree with nodes of different types (element, attribute, text etc.).
Entity identification	IRIs for globally unique identification of entities and relationships.	Use of IRIs is possible, but not enforced.
Schema	RDF-Schema and OWL allow definition of vocabularies and formal ontologies, including inference and consistency checks.	Multiple schema definition languages allow to define data types, structures and constraints.
Schema/data separation	The same representation is used for schema and data, schema reuse is strongly encouraged.	Schema can be expressed as XML. External schemata possible.
Conceptual and physical separation and model	Conceptual model only.	Conceptual model only.
Expressivity	Focus: entities and relationships forming a graph; Problematic: lists, n-ary relations, constraints, graphs	Focus: Flexible model allows representation of arbitrary data structures; Problematic: requires external knowledge for interpretation.
Data access	Linked data dereferencing (HTTP), SPARQL queries.	DOM, XQuery/XPath, Web services, files
Serialization	RDF/XML, turtle, N-Triples.	XML
World Model	Open world.	Closed world.

Ontology as a domain modelling technique assumes existence of rules that express logic in relation co-dependencies. To clarify this concept we will use a simple example. In a case of ontology modelling a domain of car-ownership might have simple rules such as "Tom owns X car" and "X is a German factory". In that case, rule inference will provide us a new knowledge saying "Tom owns German car". Rule inference engines do this automatically and can chain more than two rules and thus, provide us with more complex conclusions, resulting in more detail, clearer model of the domain.

It is important to highlight that rule inference is self-initiated process. It is a background process on all levels for all the concepts, every time that ontology is edited or the set of rules is extended. End-users do not even need to be familiar with these processes. Inference results are expressed through wider range of allowed queries for knowledge. Under certain constraints embedded in graphical interface, user can ask a direct questions and be presented with an answer as long as the answer is reachable using rule chaining.

4. SEMANTIC ENRICHMENT FRAMEWORK

The development of a semantically enriched framework for analysis of shop floor operation, with the use of state of the art tools for semantic interoperability, aims at capturing knowledge on dynamically evolving shop floor operations. Knowledge can be represented through the use of semantic models (ontologies) together with rules for knowledge extraction and inference. The Ontology Manager (OM) architecture documented on D2.2 shows the hierarchical structure of the latter, which leverages the strengths of a semantic model at different levels of abstraction, from the data-oriented ontology to the domain-specific (shop floor-oriented) ontologies. More details will be given in the next section.

Therefore the semantically enriched framework is developed with the use of open source software and tools that enable, on the one hand, the management of the semantic models and sources by leveraging the expressivity and all the characteristics of the W3C standards. On the other hand, it enables the interaction with two other SatisFactory components, i.e. Context-aware engine and the Decision Support System

4.1 ONTOLOGY MANAGER ARCHITECTURE

The SatisFactory Ontology Manager (OM) is designed as a **Multi-Layered Ontology Network** (see Figure 14). The upper element is the so-called *SatisFactory Upper Ontology*, which aims to provide a general vocabulary of the main shop floor terms, for example, *Assets*, *Procedures* and *Workers*. Such shared vocabulary is, therefore, used as a semantic bridge between domain-specific ontologies (shop-floor oriented semantic structures) and the so-called data-oriented ontologies at the very lowest level. The three *Shop Floor Procedures-Oriented Ontologies* (COMAU, SUNLIGHT, and CERTH) aim at representing the knowledge that is related to each specific industrial partner and the related Business Scenarios (BSCs), taking into consideration specific terms and elements that can be leveraged to manage and optimize human resources. Finally, the lowest layer contains the aforementioned *Shop Floor Data-Oriented Ontologies* that have been derived from the SatisFactory Common Information Data Exchange Model (CIDEM) schemas, hence, these are strictly related to the XML nature of the shop floor information flow, and aims at supporting the semantic enrichment of the CIDEM data.

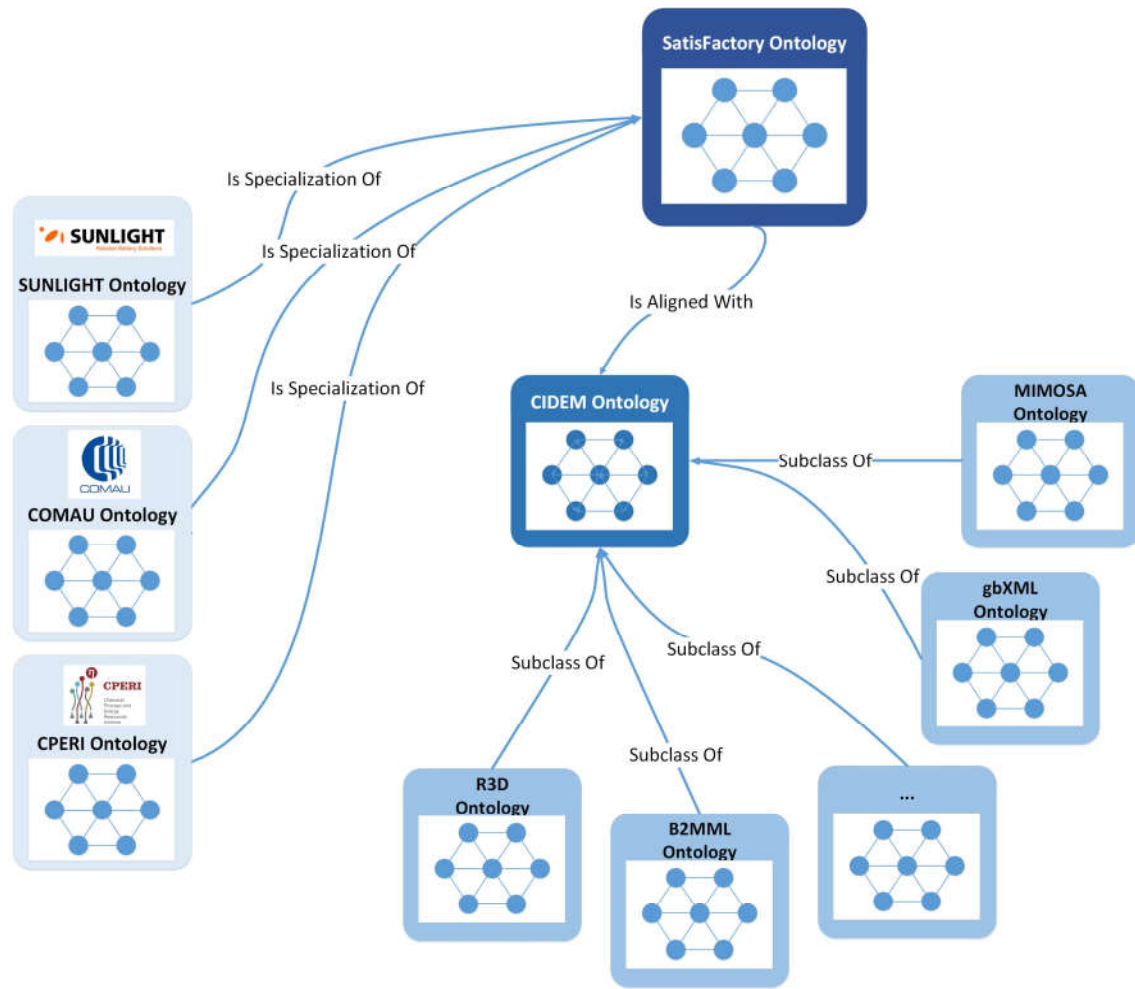


Figure 14 Ontology Manager General Architecture

4.2 USE CASES ANALYSIS

The detailed analysis of the Satisfactory Use Cases and Application Scenarios has been thoroughly presented in the deliverable D1.2. These Use Cases are developed to show the functionalities of the Satisfactory platform, and then will be used for the implementation of the selected Application Scenarios at the shop floors [12].

This sections, instead, aims to give a quick overview of the UCs that are strictly related to the Semantic Context Manager (Ontology Manager + Context-Aware Engine) in order to outline and further describe the role of the Semantic Framework in the next sections. Table 6 shows the outcome of such domain analysis. The complete description of the UCs and BSCs, instead, is thoroughly documented in D1.2.

Table 6 Analysis of the Use Cases regarding the Ontology Manager

Semantic Context Manager	Business Case Scenarios										
	1.1	1.2	2.1	3.1	4.1	4.2	4.3	5.1	5.2	5.3	6.1
UC1.1 Process for handling shop floor related information (data acquisition)	X	X	X	X	X	X		X	X	X	
UC1.2 Storage of the shop floor information and data from the multi-sensorial networks (storage of data to repository)	X	X	X	X	X	X		X	X	X	X
UC1.3 Analysis of real-time and historical info from the shop floor			X	X	X	X		X			X
UC2.1 In-factory training and support of workers using a flexible learning platform	X	X	X		X		X				
UC2.2 Validation of training actions performed at the shop floor	X		X		X		X	X	X		X
UC2.3 Presentation of the shop floor procedures utilizing heterogeneous material (work orders, manuals, schematics)	X	X	X		X		X	X		X	
UC3.1 Online recognition of workers activities	X		X	X	X	X	X	X	X	X	X
UC3.2 Incident identification based on dynamically evolving operations											X
UC3.3 Monitoring and online notification of abnormal events or alarms			X			X		X			X
UC3.4 Identification of worker's and equipment's location									X	X	X
UC4.1 Provide maintenance work plans and actions related to human-centric activities			X	X				X	X	X	
UC4.2 Acquire work schedules and sequence of	X	X	X	X	X	X	X	X	X	X	

actions											
UC4.3 Monitoring and decision support of operations and maintenance procedures	X	X	X	X	X	X	X	X	X	X	
UC4.4 Provide workers availability and allocation of resources			X	X	X	X	X	X		X	
UC4.5 Worker comfort level notifications under varying environmental conditions								X	X	X	
UC5.1 Gamification framework	X	X	X	X	X	X	X	X	X	X	
UC5.2 Platform for suggestions for improvement	X	X		X							
UC5.3 Gamified Platform for suggestions for improvement	X	X		X							
UC5.4 Instantiation of Gamification Framework							X			X	
UC6.1 Provide informative analytics using advanced visual representation								X	X		
UC6.2 Provide optimum actions and instructions with personalized dynamic information			X	X	X	X	X	X	X	X	
UC6.3 Knowledge sharing among workers based on advanced reasoning				X	X	X	X	X	X	X	
UC6.4 Presentation of shop floor information at the AR Glasses			X	X	X	X	X				

- Ontology Manager Use Case
- Semantic Context Manager Use Case

Therefore, the table above emphasizes the primitive UCs where the Semantic Context Manager will be involved and exploited for the purpose of the related BSC:

- **UC1.2** Storage of the shop floor information and data from the multi-sensorial networks (storage of data to repository)



- **UC1.3** Analysis of real-time and historical info from the shop floor
- **UC2.2** Validation of training actions performed at the shop floor
- **UC2.3** Presentation of the shop floor procedures utilizing heterogeneous material (work orders, manuals, schematics)
- **UC3.2** Incident identification based on dynamically evolving operations
- **UC4.5** Worker comfort level notifications under varying environmental conditions
- **UC6.3** Knowledge sharing among workers based on advanced reasoning
- **UC6.4** Presentation of shop floor information at the AR Glasses

4.3 ONTOLOGY REQUIREMENTS SPECIFICATIONS

With regard to the NeOn methodology, the development of semantic structures, such as the ontology models, brings the ontology designer to investigate the domain in terms end users, use cases, and requirements (functional and non-functional). These activities have been partly documented in D2.1. Here we focus on the two specific tasks:

- Support User Interaction with the DSS (ABE)
- Support User Interaction with the Context-aware engine (ISMB)

Indeed, the Ontology Manager has been developed to support the components mentioned above. Such interaction will be thoroughly described in Section 4.5 and 4.6, whereas the integration of the Semantic Framework within the SatisFactory platform is described in the next section in order to explore the way such components will communicate.

4.4 INTEGRATION TO SATISFACTORY PLATFORM

In this section the integration of the Semantic Framework within the SF platform will be investigated in terms of:

- Data transformation (RDFization)
- Data querying and extraction

4.4.1 SatisFactory data flow RDFization

The Resource Description Framework (RDF) is a family of World Wide Web Consortium (W3C⁷) specifications. This data model is based upon the idea of making statements about resources in the form of subject–predicate–object expressions. These expressions are known as triples in RDF terminology.

The SatisFactory datasets are transferred through the platform as XML, providing a so-called tree-based data model that is purely syntactic in nature. As mentioned already, and thoroughly documented in D2.2 [13], the semantic processing is standardised around RDF that provides, instead, a graph-based model. Therefore, in a transformation process we must translate the XML elements into RDF ones according to the SatisFactory ontology models. Such translation will give a valid RDF dataset that can be load in the Semantic Framework and easily processed and further analysed. The

⁷ <https://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/>

transformation process for real-time data however can be complicated because of the time and computation effort needed.

The development of the so-called data oriented ontology model (CIDEM ontology) aims at speeding up the mapping and, above all, the maintenance and reuse of these semantic structures. This wide low level ontology, in fact has been developed from the CIDEM XML Schemas, which have been analysed and then properly transformed in order to generate OWL entities (see the methodological description in D2.2).

Figure 15 shows an example of XML-RDF concepts mapping. In particular, the XML data regarding a worker (actor) are stored in the CIDEM according to the B2MML format plus ad hoc elements defined by the SF_Common Schema. Then, we will transform the following information:

(i) Actor Information, (ii) Actor ID, (iii) Actor Group, (iv) Actor Group Description, and (v) Actor Experience Level; in Object/Data properties linking the RDF subject “Actor” (or “Worker”) to the enumerated list (e.g. the actor groups are not arbitrary and are related to the worker element through an object properties) or values (e.g. worker ID is an arbitrary value linked to the worker element through a data property). Therefore we will use the following properties: (i) hasPersonnelInformation, (ii) hasActorID, (iii) hasGroupDescription, (iv) hasActorGroup, (v) hasExperienceLevel.

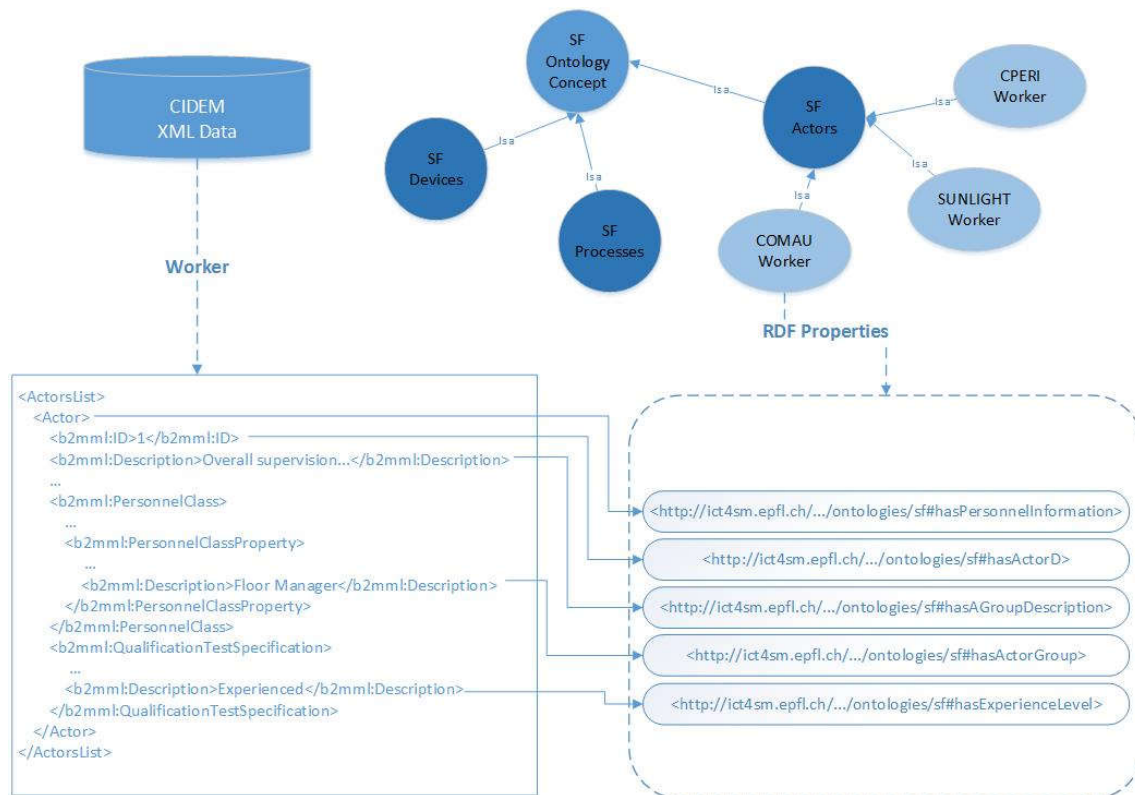
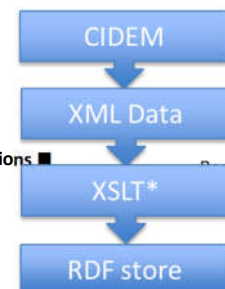


Figure 15 COMAU Worker - XML Data RDFzation

This transformation process is achieved through the use of Extensible Stylesheet Language- based transformations. XSL is basically a style sheet for



XML documents, whereas XSLT stands for XSL transformation, which allows the transformation of XML documents into other XML formats (Figure 16).

The communication between the CIDEM and the Ontology Manager, is therefore based on the following mechanism. The XML data are extracted from the CIDEM store, transformed, and then re-stored in the RDF store. Annex C introduces the XSL code example for *Work Schedule Events*

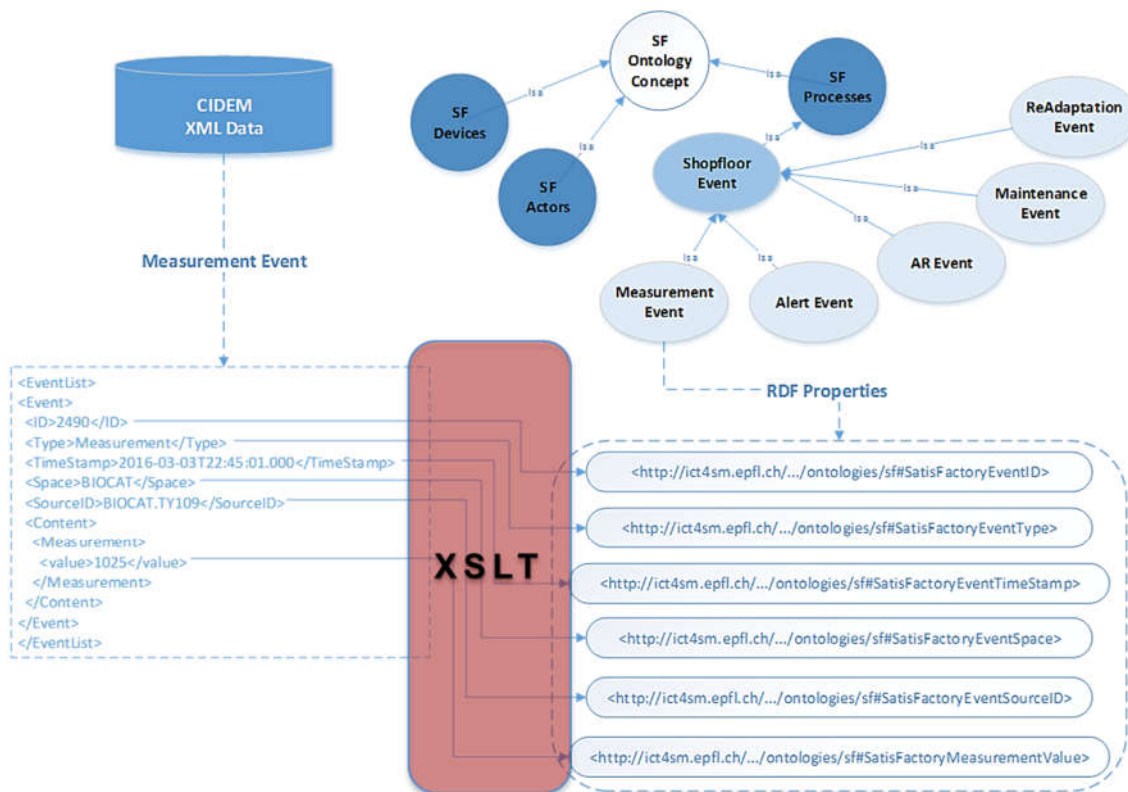


Figure 16 Use of XSL Transformation for CIDEM data RDFization

4.4.2 Data Exchange Format

The solution explored so far suggest to either use the GUI provided by the OSF to define the right Queries or create an adhoc REST services for the queries.

According to the OSF Wiki the integrated SPARQL engine can be accessed directly by using POST calls like the following, under certain restrictions (i.e. Note: All parameters have to be URL-encoded):

http://[...]/ws/sparql/?query=param1&dataset=param2&limit=param3&offset=param4&default-graph-uri=param5&named-graph-uri=param6&interface=param7&version=param8

where

- **query.** The SPARQL query to send to the Web service endpoint - param1
- **dataset.** URI referring to a target dataset to query - param2
- **limit.** Limit of the number of results to return in the resultset (max 2000) - param3

- **offset.** Offset of the "sub-resultset" from the total resultset of the query - param4
- **default-graph-uri.** Dataset to target with the sparql query (optional) -- only used for consistency with the SPARQL protocol - param5
- **named-graph-uri.** Dataset to target with the sparql query (optional) -- only used for consistency with the SPARQL protocol - param6
- **interface.** Source interface used for this web service query. The interface is a different way to process a query (different algorithms, different data management system, etc. The default interface is 'default' - param7
- **version.** (default: 3.0) Version of the interface to query - param8

Result datasets can be, then, formatted in XML or JSON (see section 2.1.5).

4.5 IDSS SUPPORT

This section describes some common scenarios regarding the Ontology Manager and the Integrated Decision Support System. It is a base for a common understanding of their interaction in the framework of Satisfactory project. An initial description of the information flow is given (see Figure 17) together with a thorough description of the aforementioned common scenarios. Therefore, a description of how the Semantic Framework is going to be used to support the complex decision process of the DSS within the extensive Satisfactory Framework will be presented in the next paragraphs.



Figure 17 iDSS – OM interaction

Basically, the DSSCore is the component that provides the main decision support functionality of iDSS. It connects the various data and transforms them into actionable knowledge (decision). In the base scenario (see Figure 18) the DSSCore responds to an external incident of a malfunction with a proper work order and assigns the task to an appropriate worker. More details are documented in D3.5 [14].

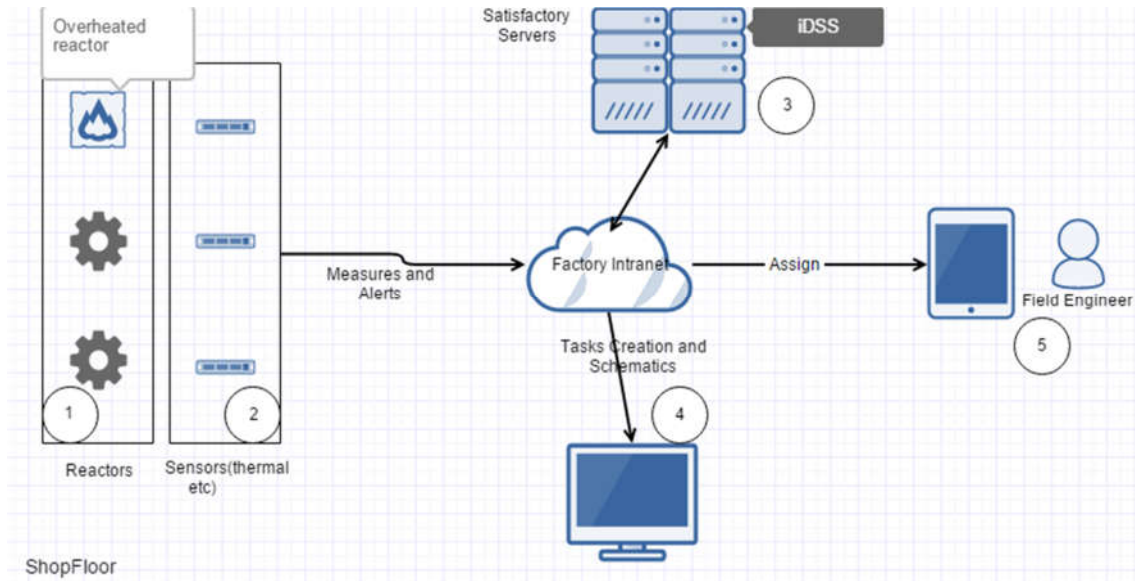


Figure 18 DSS responds to external Alert

However, in this scenario the iDSS can use the ontology framework to enrich the response with the most accurate data prediction, based on the classification that the Ontology Manager provides. Specifically, the iDSS needs to estimate the task duration and to assign it to the proper worker according to the expertise and level of experience in order to create the work order. Moreover, in order to properly estimate the task duration, it is useful to know the duration of previous tasks related to specific assets (or area). For a specific category of tasks, hence, the entities that would be necessary for the iDSS can be summarized as in Table 7.

Table 7 Entities needed for DSS workers classification

Entity	Comments
Worker Group	The group that worker belongs (e.g. Process Technician, Process Supervisor, IT Technician)
Worker Skills	The level of the experience of the worker for a given worker group can be expressed with a discrete value (Trainee, Novice, Experienced). However, a worker can be suitable for enlisted into more than one groups. This value can be expressed as a percentage of suitability of such worker for a task that is supposed to be performed by a different worker group.
Task Duration	This entity must provide the task duration median and the worst case for a specific asset and a specific task group. Using that information, iDSS estimates the duration for the given work order and maybe assert the performance.

Once the Ontology Manager analyses the information below and give to the iDSS the requested data results, the latter can take the following steps.

1. Creates the Work Order.



2. Estimates the duration according to specified duration and the information from OM.
3. Picks the right worker group and experience level, according to the specified procedure.
4. Sends the information to the Re-Adaption toolkit to assign the worker and re-schedule production.

Therefore, one of the core activities of this component is to manage the human resources and the tasks that are daily performed at the shop floor level. These will be partly supported by the Ontology Manager, which aims at providing semantically-enriched information, and then aggregate the information flow in a way that data together with their semantics can be exchanged within the SatisFactory ecosystem by enhancing the decision and analysis process.

Another important operation of DSS is the everyday scheduling, i.e. the daily appointment of each task to specific worker. The OM can enrich the procedure by providing the correct context for each job, so each worker is assigned the right level of task difficulty (not a very difficult or a very easy and mundane). For this scenario iDSS uses the same entities described above according to the following steps:

1. Classifies the tasks to worker groups and levels.
2. Uses the entities of OM to classify the workers.
3. Assigns the workers to each task classification.
4. Produces the schedule for each group.
5. Combines all the separated schedules into one.
6. Publishes the current schedule into CIDEM.

In these context, the analysis of the information flow through the exploitation of the SPARQL query engine, which deals with the enriched structure and content of the SatisFactory data, represents the key aspect of interaction between the Ontology Manager and the iDSS. According to the DoW and the requirement specifications, three application cases have been agreed in order to explore such interaction. Moreover, it is worth pointing out that the aforementioned term “applications cases” must be not confused with the Application scenarios, Business Case Scenarios and Use Cases that have been described in D1.2. Three specific analysis will be, performed through the semantic framework:

- Task duration analysis
- Worker suitability analysis
- Experience/Worker Group based analysis

For the applications above, the Semantic Framework needs an oversight of history activities, context, and other shop floor data which will be, in fact, managed, analyzed and then given as input to the iDSS in a way that can be further exploited to enhance the decisis process.

Following, the applications presented above will be further described in terms of:

- *Input data/information*, which will finally result in a set of elements of the SPARQL query.
- *Semantic Data* that refers to the CIDEM xml data that will be RDFized according to the ontology models, and then, further analyzed according to specific needs and queries.
- *Expected Outcome(s)* of the query, which is basically described by content, semantic and format.

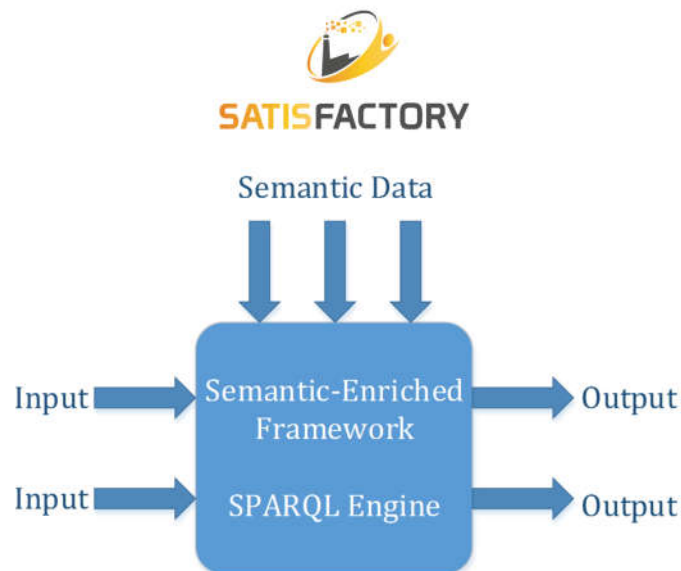


Figure 19 Black Box: Application examples

The same representation will be used to present any other kind of application that involves the use of the SPARQL query engine to explore and browse across the semantics-enriched data stored in the Semantic Framework RDF store.

Lastly, the information used in Section 4.5 and 4.6 have been gathered from CPERI shop floor in the last months preceding the submission of this deliverable. Even if we mainly focused our effort on the definition of methodologies and common implementation cases, these take an additional value from the validation phase performed through real shop floor information and data.

4.5.1 Task duration analysis

The interaction of the iDSS and the Semantic Framework is investigated through the provision of the aforementioned specific analysis, whose results will be provided to the former component according to a Satisfactory semantics-enriched format (RDF). The following scenario partly meets the UCs list presented in Section 4.2. In fact, it is justified by UC-1.3, UC-2.3, UC-4.5, and UC-6.3.

According to the representation in Figure 20, an overview of the information/data flow that is required to describe such example is following presented.

The task duration analysis implies the exploration of diverse information sources, such as historical records of task performance, task information (estimated duration, expected worker group, expected experience, work area, etc.). Most of these data might be retrieved in the CIDEM, although they require a preliminary semantic enrichment, prior loading in the triple store (see Section 4.4). However, concepts such as *expected worker group* and *experience* are introduced by the Semantic Framework, aiming to make explicit such tacit shop floor knowledge. Therefore, the task duration analysis report will show as outcome the Task ID (and/or Description), the Median Value, the Worst Case, and then, it will be filtered by worker groups, assets, and work area.

Input	<ul style="list-style-type: none"> Task ID Time frame (optional)
Semantic Data	<ul style="list-style-type: none"> Task estimated duration Task expected worker group

	<ul style="list-style-type: none"> • Task expected experience • Task performance records • Worker group • Worker experience • Work Areas
Expected Outcome (filtered by Task, Worker groups, Assets, Work area)	Analysis of the performances <ul style="list-style-type: none"> • Median value • (Best and)Worst Cases

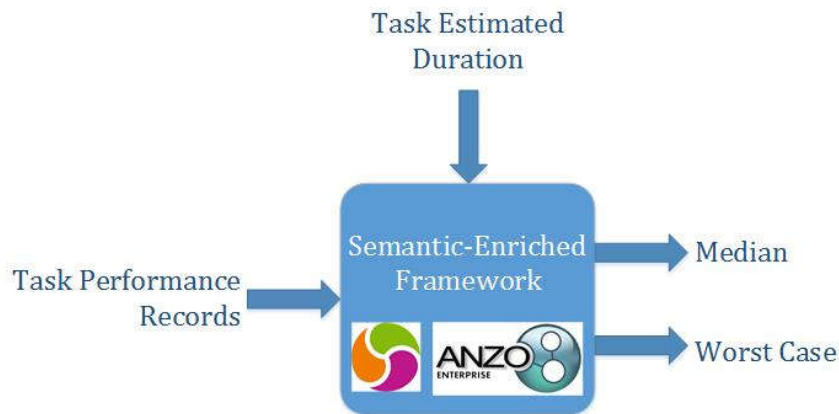
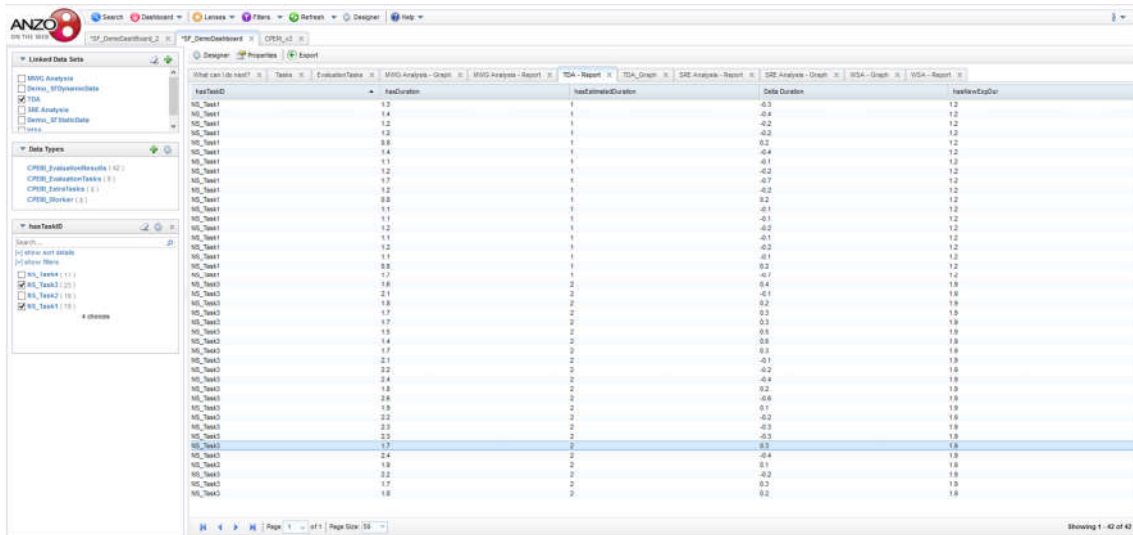


Figure 20 Black Box: Task Duration analysis

Figure 21 shows the Task Duration Analysis (TDA) report that is obtained through the use of the (commercial) software ANZO Enterprise. Such report, show the actual duration, the expected one and the delta between these two value regarding all the activities performed in a specific time frame (e.g. the last 2 months).



TaskID	TaskDuration	TaskEstimatedDuration	Delta Duration	TaskNameExpDur
NS_Task1	1.2	1	-0.3	1.2
NS_Task1	1.4	1	-0.4	1.2
NS_Task1	1.2	1	-0.2	1.2
NS_Task1	1.2	1	-0.3	1.2
NS_Task1	0.8	1	0.3	1.2
NS_Task1	1.4	1	-0.4	1.2
NS_Task1	1.1	1	-0.1	1.2
NS_Task1	1.2	1	-0.2	1.2
NS_Task1	1.7	1	-0.7	1.2
NS_Task1	1.2	1	-0.2	1.2
NS_Task1	0.8	1	0.3	1.2
NS_Task1	1.1	1	-0.1	1.2
NS_Task1	1.1	1	-0.1	1.2
NS_Task1	1.2	1	-0.2	1.2
NS_Task1	1.1	1	-0.1	1.2
NS_Task1	1.2	1	-0.3	1.2
NS_Task1	1.1	1	-0.1	1.2
NS_Task1	0.8	1	0.3	1.2
NS_Task1	1.7	1	-0.7	1.2
NS_Task2	1.8	2	-0.4	1.8
NS_Task2	2.1	2	-0.1	1.8
NS_Task2	1.7	2	0.3	1.8
NS_Task2	1.7	2	0.3	1.8
NS_Task2	1.9	2	-0.1	1.8
NS_Task2	1.8	2	-0.2	1.8
NS_Task2	2.4	2	-0.4	1.8
NS_Task2	1.8	2	-0.2	1.8
NS_Task2	1.8	2	-0.6	1.8
NS_Task2	1.9	2	0.1	1.8
NS_Task2	2.2	2	-0.2	1.8
NS_Task2	2.3	2	-0.3	1.8
NS_Task2	2.5	2	-0.5	1.8
NS_Task2	1.7	2	0.3	1.8
NS_Task3	2.4	2	-0.4	1.8
NS_Task3	1.8	2	-0.1	1.8
NS_Task3	2.2	2	-0.2	1.8
NS_Task3	1.7	2	0.3	1.8
NS_Task3	1.8	2	-0.2	1.8

Figure 21 TDA report - ANZO

Figure 22 and Figure 23, instead, show the graph representation of the results reported above. The end users can set customized filter to ease the visualization of the results in both presented platforms (Anzo and OSF). In particular, here we implemented a *Tasks filter* that allows the visualization of the data regarding a specific (non scheduled) task, e.g. the results in Figure 24 are presented through a multi-container ANZO dashboard by selecting the Task 1 analysis results.

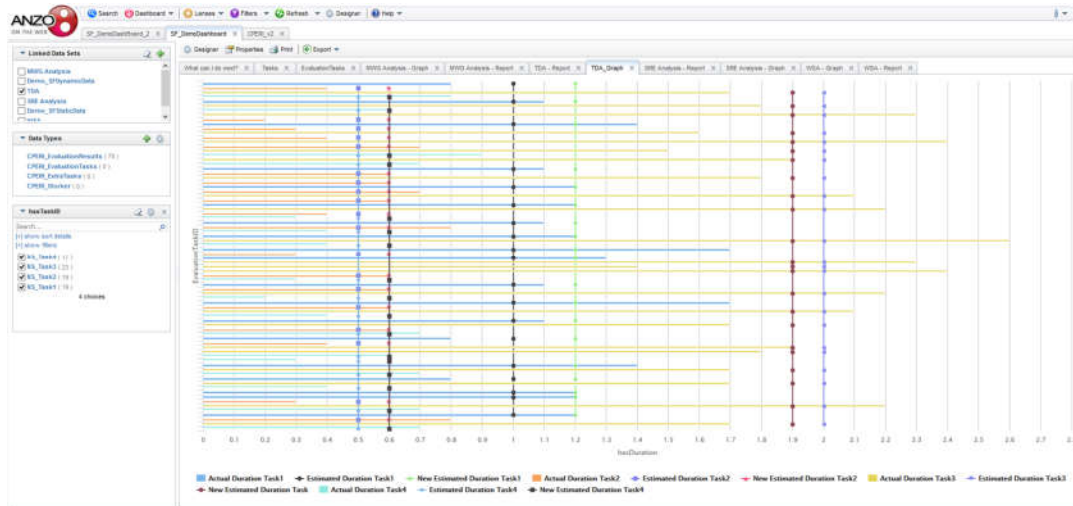


Figure 22 TDA graph – ANZO



Figure 23 TDA graph - OSF

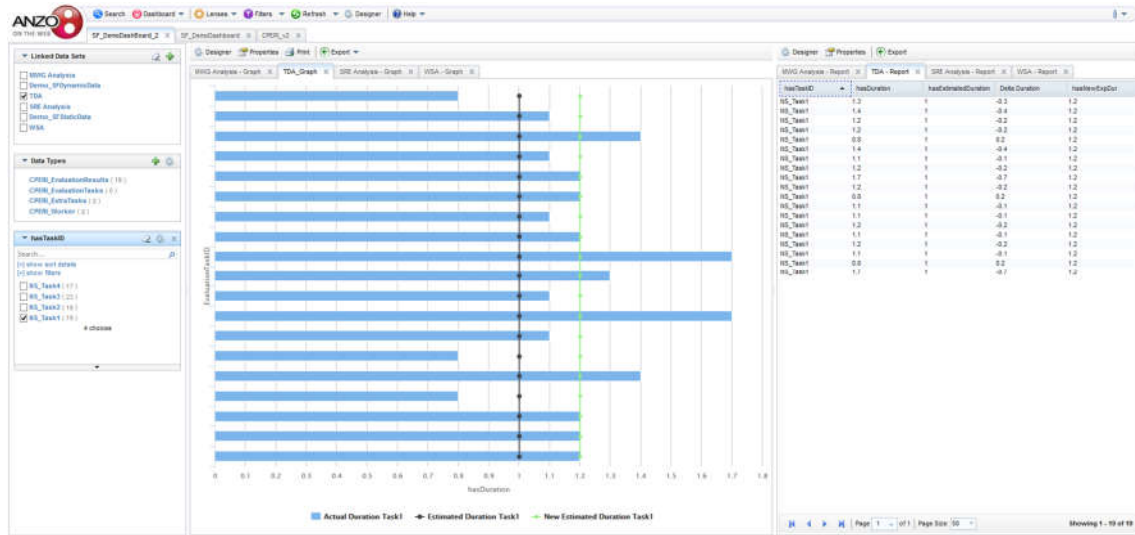


Figure 24 Task Duration Analysis example - Task 1

The figure above, in particular, shows one possible use of such results. The mean actual duration is calculated and visualized (green vertical line in Figure 24) in order to give to the downstream component (in general, the end user) an indication regarding the misevaluation of the expected duration of the analysed task, together with its potential modification in line with the new evaluation.

The exploitation of the networked nature of the semantics-enriched data flow enables the definition of diverse types of filters and evaluations, for example, taking into account the full context of a specific task (area, assets, etc.) and the features of the worker that performed it (WG, skills, age, etc.). This wider information set might be leveraged to perform more refined assessments based on the weighted impact of the aforementioned factors.

4.5.2 Worker suitability analysis

The application presented in this section matches most of the UCs documented in Section 4.2. In fact, it is justified by UC-1.3, UC2.2, UC-2.3, UC-4.5 UC-6.3. Through the following application we aim to perform an analysis of the workers suitability for the tasks that might be triggered from the system in order to meet extra ordinary shop floor requests. This example, in fact, focuses on the so-called extra tasks that have defined by CERTH/CPERI. However, with necessary expedients, this approach can be easily extended to scheduled tasks and, obviously, adapted to any other SatisFactory shop floors by leveraging the other shop-floor oriented (domain specific) semantic structures.

Unlike the previous example, here the focus was shifted on the worker profiles rather than the sole tasks. In particular, the Semantic Framework introduces some enriched elements information, such as the MWG (*matchability* index for worker group), SRE (*suitability* index for requested experience), together with the Expected Worker Group and the Expected Experience Level for a specific task, just like the previous example.

Input	<ul style="list-style-type: none"> • Task ID • Work Area (optional)
Semantic Data	<ul style="list-style-type: none"> • Worker group • Worker experience • MWG • SRE • Expected worker group • Expected experience
Expected Outcome	Analysis of workers suitability for each task

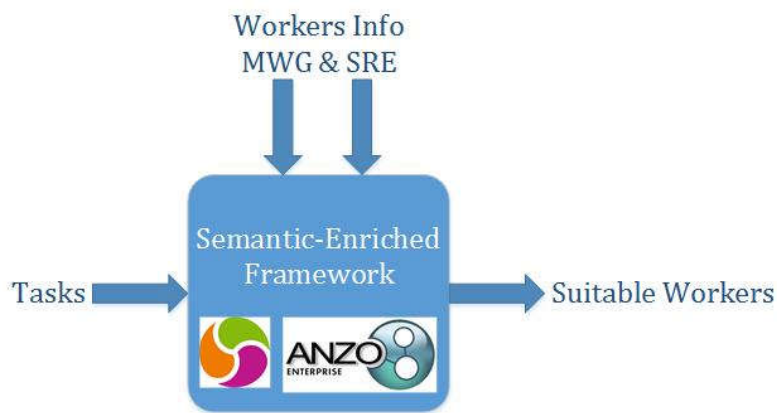


Figure 25 Black Box: Worker Suitability Analysis

The MWG and SRE indexes aim to ease the decision process related to the choice of a worker group that is different from the expected one for a specific task. In particular, we start from the assumption that each worker group can be further discerned, and then characterized, by a set of skills. A the detailed description of the worker groups has been carried out in the framework of D1.2 and summarized also in the Annex A of this deliverable (see Table 15). Such analysis allows the extraction of “standard” skills that can be used to describe each specific worker group (see Table 8).

Table 8 List of workers skills

Set of Skills	
Skill 1.	Decision making
Skill 2.	Knowledge on chemical processes
Skill 3.	Management of human resources
Skill 4.	Administrative skills
Skill 5.	Management of shop floor
Skill 6.	Able to collaborate and coordinate actions
Skill 7.	Time Management Skills
Skill 8.	Knowledge on pilot plant operations
Skill 9.	Able to use monitoring systems
Skill 10.	Management of the operations of pilot plants
Skill 11.	Automation, electrical and mechanical knowledge
Skill 12.	Maintenance of the shop floor facilities
Skill 13.	Able to respond fast and efficient to incidents
Skill 14.	Maintenance of the pilot plants
Skill 15.	Knowledge of mechanical design software
Skill 16.	Preserve information related to performed procedures
Skill 17.	Knowledge on automation software
Skill 18.	Able to use tools for construction of Pilot Plants
Skill 19.	Able to handle malfunctions
Skill 20.	Sharing information ability
Skill 21.	Knowledge and maintenance of IT infrastructures
Skill 22.	Knowledge on electrical systems
Skill 23.	Able to understand electrical schematics and manuals
Skill 24.	knowledge of electrical design software
Skill 25.	Able to recognize and report malfunction
Skill 26.	knowledge of safety measures on Pilot Plants operation

Therefore, each worker group will be assigned a different set of skills (see Table 9). This way we can define a skills-based description of the worker groups that will guide, on the one hand, the definition of rules that enable inference-based validation of the worker groups within the semantic models. On the other hand, it allows the set-up of initial values for the MWG index and, at the same time, a base for its continuous adjustment and evaluation.

Table 9 List of skills per worker group

Group of actors related with the set of skills
A - Floor Manager - (1, 2, 3, 4, 5, 6)
B - Process Supervisor - (1, 2, 3, 4, 7, 8, 9, 10)
C - Maintenance Manager - (5, 6, 7, 11, 12, 20)
D - Maintenance Supervisor - (3, 6, 8, 11, 13, 20)
E - Process Operator - (6, 7, 9, 13, 18, 25,26)
F - Process Technician - (6, 13, 14, 17, 18, 19)
G - Electrical Technician - (6, 13, 14, 19,22, 23, 24)
H - Control Automation - (6, 7, 13, 17, 19, 21)
I - IT Technician - (6, 7, 13, 17, 19, 21)

The introduction of these information within the Semantic Framework might also enable the validation and classification of the worker's profile according to his/her skills rather than his/her "title". From a semantic-based reasoning point of view, in fact, we might state the following axioms:

*hasSkill some Skill **and** hasSkill **only** (Skill1, Skill2, Skill3, Skill4, Skill5, Skill6) →
hasWorkerGroup(Worker,FloorManager)*

*hasSkill some Skill **and** hasSkill **only** (Skill1, Skill2, Skill3, Skill4, Skill7, Skill8, Skill9, Skill10) →
hasWorkerGroup(Worker,ProcessSupervisor)*

*hasSkill some Skill **and** hasSkill **only** (Skill5, Skill6, Skill7, Skill11, Skill12, Skill20) →
hasWorkerGroup(Worker,MaintenanceManager)*

*hasSkill some Skill **and** hasSkill **only** (Skill3, Skill6, Skill8, Skill11, Skill13, Skill20) →
hasWorkerGroup(Worker, MaintenanceSupervisor)*

*hasSkill some Skill **and** hasSkill **only** (Skill6, Skill7, Skill9, Skill13, Skill18, Skill25, Skill26) →
hasWorkerGroup(Worker,ProcessOperator)*

*hasSkill some Skill **and** hasSkill **only** (Skill6, Skill13, Skill14, Skill17, Skill18, Skill19) →
hasWorkerGroup(Worker,ProcessTechnician)*



*hasSkill some Skill **and** hasSkill **only** (Skill6, Skill13, Skill14, Skill19, Skill22, Skill23, Skill24) →
hasWorkerGroup(Worker, Electrical Technician)*

*hasSkill some Skill **and** hasSkill **only** (Skill6, Skill7, Skill13, Skill17, Skill19, Skill21) →
hasWorkerGroup(Worker, Control Automation)*

*hasSkill some Skill **and** hasSkill **only** (Skill6, Skill7, Skill13, Skill17, Skill19, Skill21) →
hasWorkerGroup(Worker, ITTechnician)*

That said, the *matchability* index values regarding each worker group are extracted from a joint analysis of the aforementioned skill-based description of the worker group and an experience-based analysis of the shop floor workers. Table 10 shows such values. The meaning of this index can be summarized as follows, if Task_x requires the Worker Group “H”, but the only available Worker Groups are “B”, and “G”, then such two groups will match the task assignment in the following way:

- Worker Group B is 20% suitable for the Task_x
- Worker Group G is 50% suitable for the Task_x

This is because, group “B” has 1 out of 6 “H” skills (16,7%), while group “G” has 3 out of 6 “H” skills (50,0%). This skill-based evaluation is then adjusted according to the expert experience by creating a solid contribution to the shop floor knowledge base that resides in the Semantic Framework.

Table 10 MWG indexes

		Worker Group								
		A	B	C	D	E	F	G	H	I
hasMatchabilityIndex_with	A	1.00	0.8	0.8	0.6	0.4	0.2	0.2	0.4	0.3
	B	0.6	1.00	0.5	0.3	0.9	0.4	0.1	0.2	0.1
	C	0.7	0.6	1.00	0.9	0.3	0.6	0.6	0.5	0.2
	D	0.7	0.6	0.9	1.00	0.3	0.6	0.6	0.5	0.2
	E	0.0	0.7	0.1	0.0	1.00	0.2	0.1	0.1	0.1
	F	0.1	0.3	0.3	0.2	0.7	1.00	0.6	0.5	0.2
	G	0.1	0.2	0.2	0.2	0.4	0.8	1.00	0.5	0.2
	H	0.1	0.2	0.2	0.2	0.5	0.9	0.9	1.00	0.60
	I	0.1	0.1	0.2	0.1	0.3	0.6	0.6	0.6	1.00

Much the same applies to the *Suitability* index for Requested Experience (SRE), which aims to define the suitability of a specific level of experience while performing a definite task (see Table 11). This doesn't take into consideration skills or specific capabilities. Indeed, it is mostly related to the level of difficulty that can be assigned to the task, and indirectly related to the work force costs that the company might be keen to allocate on that specific task.

Table 11 SRE Indexes

Task ID	SRE Experienced	SRE Novice	SRE Trainee	Task ID	SRE Experienced	SRE Novice	SRE Trainee
1	1	0.4	0	13	0.3	1	0.4
2	0.7	1	0.3	14	0.5	1	0.2
3	1	0.4	0	15	1	0.6	0.2
4	0.2	1	0.7	16	1	0.7	0
5	1	0	0	17	1	0	0
6	1	0.2	0	18	1	0.5	0
7	1	0	0	19	1	0.7	0.3
8	1	0.3	0	20	1	0	0
9	1	0.4	0	21	0	1	1
10	1	0.7	0.4	22	1	1	0.6
11	1	0.6	0	23	1	0.6	0
12	1	0.5	0	-	-	-	-

The ability of retrieving and manipulating the semantic enriched datasets that are stored in the RDF (triples) store is achieved through the use of the SPARQL Protocol and RDF Query Language (in short, SPARQL). This is, in fact, a semantic query language that enhance the system with the capabilities mentioned above. On the Semantic Framework, the worker suitability analysis report is then obtained through the execution of a SPARQL query that goes through the shop floor linked data and combines the effect of the MWG and SRE indexes (see Figure 27).



Figure 26 Worker Suitability Analysis – Virtuoso SPARQL Query Editor



The screenshot shows the results of the SPARQL query in a table format. The table has several columns: 'Task', 'RAG', 'REL', 'ActorID', 'Group', 'Experience', 'HNG', 'Index', 'SRE', 'Index', 'Suitability'. Each row represents a worker's suitability for a specific task. The 'Task' column contains task IDs and names. The 'RAG' column contains RAG IDs and names. The 'REL' column contains relationship IDs and names. The 'ActorID' column contains actor IDs and names. The 'Group' column contains group IDs and names. The 'Experience' column contains experience levels and names. The 'HNG' column contains HNG IDs and names. The 'Index' column contains index IDs and names. The 'SRE' column contains SRE IDs and names. The 'Index' column contains index IDs and names. The 'Suitability' column contains suitability scores.

Figure 27 Worker Suitability Analysis - SPARQL query results

The query that is shown in Figure 26 produces a result dataset that can be serialized in different formats: HTML, Spreadsheet, XML, JSON, Javascript, Turtle, RDF/XML, N-Triplets, CSV, TSV.

The visualization of such information, even if it is not strictly requested in the description of T3.1, has been taken into consideration as one of the exploitation facets documented in this deliverable. Indeed, Figure 28 shows the WSA as it is obtained through the use of ANZO. The information presented on the dashboard (graph representation + results report) allows the user to get a detailed overview of the most likely worker (or most suitable worker) that should perform all the task taken into consideration during the analysis process. Obviously the calculation effort increases according to the amount of information required to perform the analysis. However, the implementation tests

performed so far on a local machine showed reasonable timing to execute the suitability analysis of 20 workers for 4 tasks.

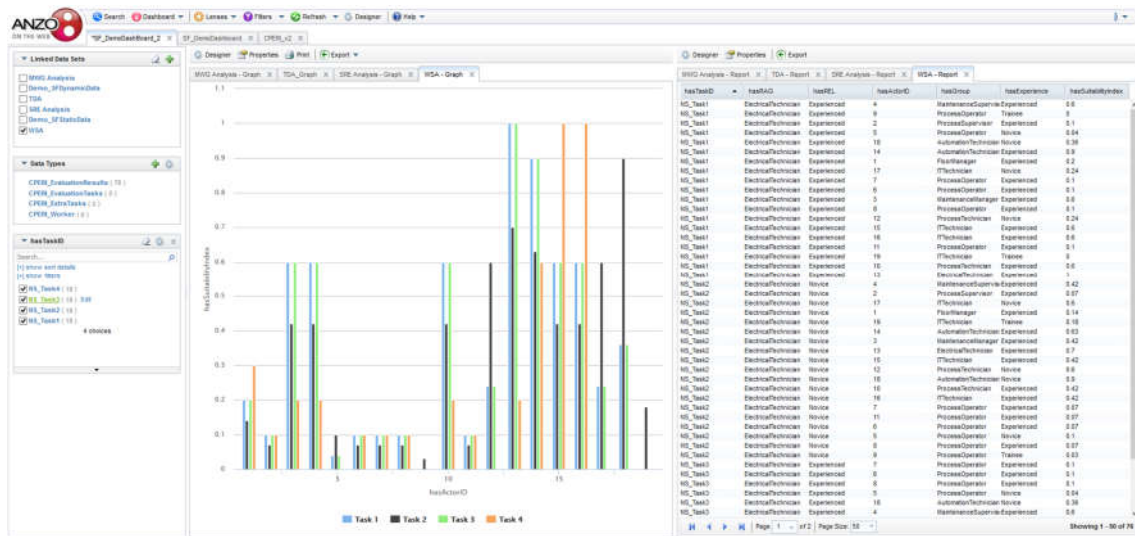
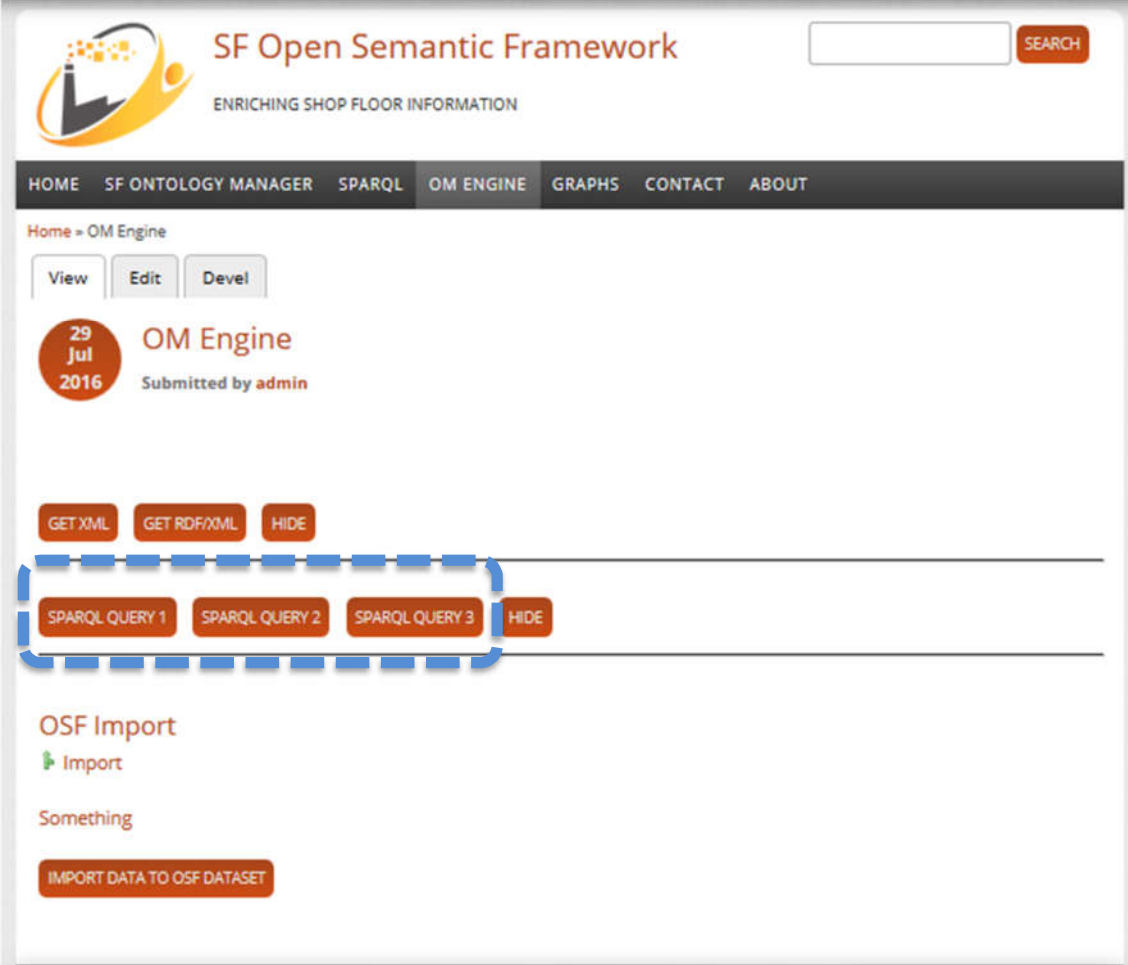



Figure 28 Worker Suitability Analysis on ANZO

Much the same applies for the results obtained through the Semantic Framework (deployed with OSF), on which graph representation and results reporting features has been implemented as well. The Satisfactory Semantic Framework, in fact, allows the end users to perform such preset SPARQL query-based analysis (see Figure 29) or customize it through the SPARQL Query Engine (see Figure 30), and then, obtain the results datasets in any of the aforementioned xml-serialized formats (Figure 27 and Figure 31 show the HTML format, emphasizing the enriched nature of the results datasets). These results are, therefore, can be further investigated through the user friendly graph tab of the Semantic Framework which, moreover, allows the filtering of the information similarly to ANZO (see Figure 32).



The screenshot displays the 'SF Open Semantic Framework' web application. The header includes the application logo, title, and a search bar. A navigation menu contains links for HOME, SF ONTOLOGY MANAGER, SPARQL, OM ENGINE (selected), GRAPHS, CONTACT, and ABOUT. The main content area shows the 'OM Engine' section, submitted by 'admin' on '29 Jul 2016'. It features buttons for 'View', 'Edit', and 'Devel'. Below this, there are buttons for 'GET XML', 'GET RDF/XML', and 'HIDE'. A dashed blue box highlights a row of buttons: 'SPARQL QUERY 1', 'SPARQL QUERY 2', 'SPARQL QUERY 3', and 'HIDE'. Further down, the 'OSF Import' section includes an 'Import' button and a 'Something' section with an 'IMPORT DATA TO OSF DATASET' button.

Figure 29 Pre-set queries – Sematic Framework



SF Open Semantic Framework

ENRICHING SHOP FLOOR INFORMATION

[HOME](#)
[SF ONTOLOGY MANAGER](#)
[SPARQL](#)
[OM ENGINE](#)
[ABOUT](#)
[CONTACT](#)

Home » SPARQL

7 May 2016

SPARQL

Submitted by admin

The SPARQL Web service is used to send custom SPARQL queries against the OSF Web Service data structure. This is a general purpose querying Web service.

Virtuoso SPARQL Query Editor

[About](#) | [Namespace Prefixes](#) | [Inference rules](#)

Default Data Set Name (Graph IRI)

Query Text

```

PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX sf: <http://ict4sm.epfl.ch/files/content/sites/licp/files/ontologies/sf#>
SELECT DISTINCT ?TaskID ?Group ?RAG ?Count ?tot ((xsd:double(?Count)/ xsd:double(?tot)) AS
?Percent) ?MWG
WHERE {
    ?EvTask a sf:CPERI_EvaluationTasks . ?ExTask a sf:CPERI_ExtraTasks . ?ExTask sf:hasTaskID
?TaskID . ?ExTask sf:hasRequiredActorGroup ?RAG . ?EvTask sf:hasTaskID ?TaskID . ?EvTask
sf:hasActorID ?ActID . ?Actor a sf:CPERI_Worker . ?Actor sf:hasActorID ?ActID . ?Actor
sf:hasActorGroup ?Group .
    ?RAG sf:hasMWGIndexWith_FloorManager ?MWG_FM . ?RAG
sf:hasMWGIndexWith_ProcessSupervisor ?MWG_PS . ?RAG sf:hasMWGIndexWith_MaintenanceManager
?MWG_MM . ?RAG sf:hasMWGIndexWith_MaintenanceSupervisor ?MWG_MS . ?RAG
sf:hasMWGIndexWith_ProcessOperator ?MWG_PO .
    ?RAG sf:hasMWGIndexWith_ProcessTechnician ?MWG_PT . ?RAG
sf:hasMWGIndexWith_ElectricalTechnician ?MWG_ET . ?RAG
sf:hasMWGIndexWith_AutomationTechnician ?MWG_AT . ?RAG sf:hasMWGIndexWith_ITTechnician
?MWG_IT .
    BIND( IF(?Group= sf:FloorManager, ?MWG_FM, IF(?Group= sf:ProcessSupervisor,
?MWG_PS, IF(?Group= sf:MaintenanceManager, ?MWG_MM, IF(?Group= sf:MaintenanceSupervisor ,
?MWG_MS, IF(?Group= sf:ProcessOperator, ?MWG_PO, IF(?Group= sf:ProcessTechnician, ?MWG_PT, IF(?Group= sf:ElectricalTechnician, ?MWG_ET, IF(?Group= sf:AutomationTechnician, ?MWG_AT, IF(?Group= sf:ITTechnician, ?MWG_IT, ?MWG ))))))) AS ?MWG )
}

```

Sponging:

Results Format:

Execution timeout: milliseconds (values less than 1000 are ignored)

Options: ☒ Strict checking of void variables

(The result can only be sent back to browser, not saved on the server, see [details](#))

Copyright © 2016 Openlink Software

Figure 30 OM - SPARQL Query Engine – Semantic Framework



SF Open Semantic Framework

ENRICHING SHOP FLOOR INFORMATION

HOME SF ONTOLOGY MANAGER SPARQL OM ENGINE ABOUT CONTACT

Home » SPARQL

View Edit Devel

7
May
2016

SPARQL

Submitted by admin

The SPARQL Web service is used to send custom SPARQL queries against the OSF Web Service data structure. This is a general purpose querying Web service.

TaskID	Group	RAG	Count	tot	Percent	MWG
"NS_Task1"^^<http://www.w3.org/2001/XMLSchema#string>	http://ict4sm.epfl.ch/files/content/sites/licp/files/ontologies/sf#ProcessOperator	http://ict4sm.epfl.ch/files/content/sites/licp/files/ontologies/sf#ElectricalTechnician	7	31	0.225806	"0.1"
"NS_Task1"^^<http://www.w3.org/2001/XMLSchema#string>	http://ict4sm.epfl.ch/files/content/sites/licp/files/ontologies/sf#FloorManager	http://ict4sm.epfl.ch/files/content/sites/licp/files/ontologies/sf#ElectricalTechnician	2	31	0.0645161	"0.2"
"NS_Task1"^^<http://www.w3.org/2001/XMLSchema#string>	http://ict4sm.epfl.ch/files/content/sites/licp/files/ontologies/sf#ProcessSupervisor	http://ict4sm.epfl.ch/files/content/sites/licp/files/ontologies/sf#ElectricalTechnician	1	31	0.0322581	"0.1"
"NS_Task1"^^<http://www.w3.org/2001/XMLSchema#string>	http://ict4sm.epfl.ch/files/content/sites/licp/files/ontologies/sf#MaintenanceSupervisor	http://ict4sm.epfl.ch/files/content/sites/licp/files/ontologies/sf#ElectricalTechnician	3	31	0.0967742	"0.6"
"NS_Task1"^^<http://www.w3.org/2001/XMLSchema#string>	http://ict4sm.epfl.ch/files/content/sites/licp/files/ontologies/sf#AutomationTechnician	http://ict4sm.epfl.ch/files/content/sites/licp/files/ontologies/sf#ElectricalTechnician	1	31	0.0322581	"0.9"
"NS_Task1"^^<http://www.w3.org/2001/XMLSchema#string>	http://ict4sm.epfl.ch/files/content/sites/licp/files/ontologies/sf#ProcessTechnician	http://ict4sm.epfl.ch/files/content/sites/licp/files/ontologies/sf#ElectricalTechnician	2	31	0.0645161	"0.6"

Figure 31 WSA results - Semantic Framework

Select Graph

WSA

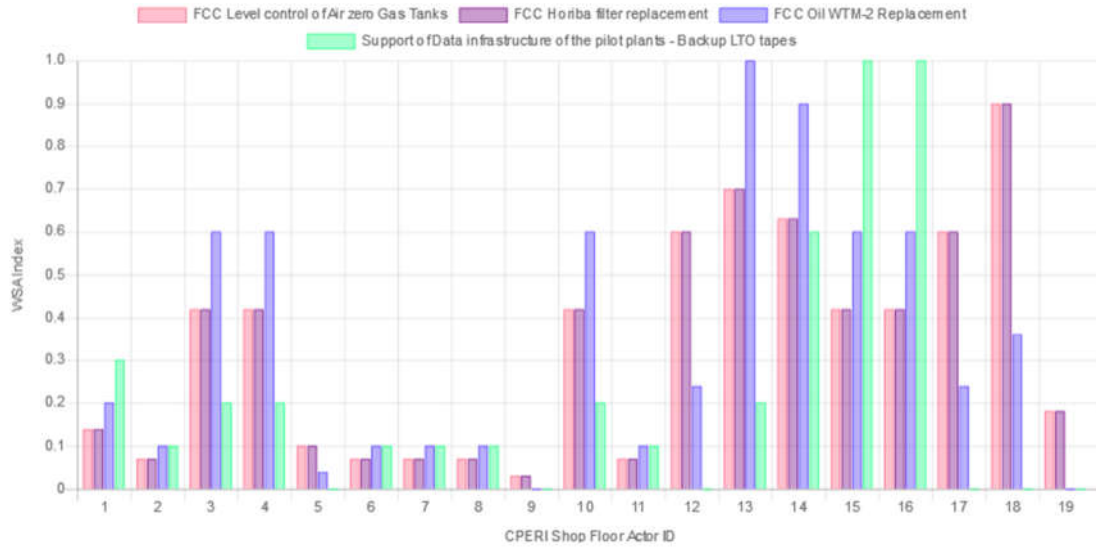


Figure 32 WSA graph - Semantic Framework

4.5.3 Experience/Worker Group based analysis

The Ontology Manager can provide a valuable context for the data analytics part of iDSS. Specifically, the OM can provide information about the workers and the utilization of the work force for a shop floor with particular emphasis on the worker profiles and task characteristics. This section aims to provide a description of the methodology adopted to continuously evaluate and, eventually, readapt two key entities that play an important role in the two analysis mentioned earlier. These entities are detailed below (see Table 12).

Table 12 SRE and MGW analysis fields

Entity	Fields	Notes
Experience-based Tasks analysis	<ul style="list-style-type: none"> • Task • Actual Experience level • Expected experience level • Expected SRE Index • New (suggested) SRE Index 	<p>Both entities can be filtered by specific groups or tasks to ease, for example, the visualization of the results.</p> <p>The new SRE index is calculated by investigating the experience of the actual worker performing the sample tasks.</p>
Worker Group-based Tasks analysis	<ul style="list-style-type: none"> • Task • Actual Actor Group that performed the evaluated task • Requested (most suitable) Actor Group for the evaluated task • Number of times the Actual Actor Group performed the task • Size of the evaluation sample • Expected MGW index • New (suggested) MGW index 	<p>The new MGW index is calculated as the percentage of workers (belonging to the non-requested worker group) performing the evaluated task and grouped by worker groups.</p>

Therefore, the end user might see how the workforce can be distributed across the analyzed tasks, and if there exist some needs or lack that should be further investigated. Figure 33 and Figure 34 show the implementation through ANZO Enterprise that is similar to the the implementation on the SatisFactory Semantic Framework (see Figure 35Figure 36).

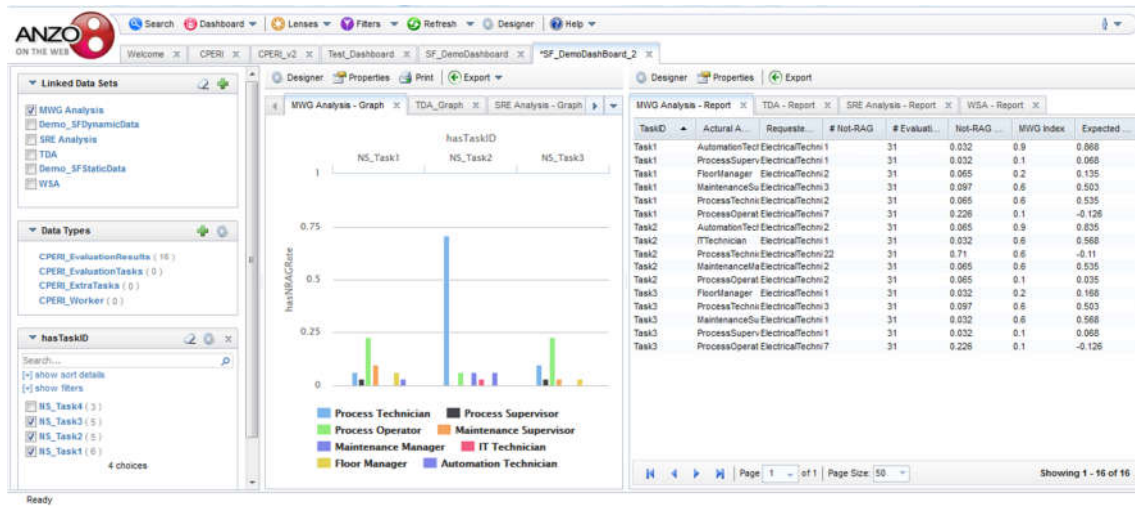


Figure 33 MWG Analysis - ANZO

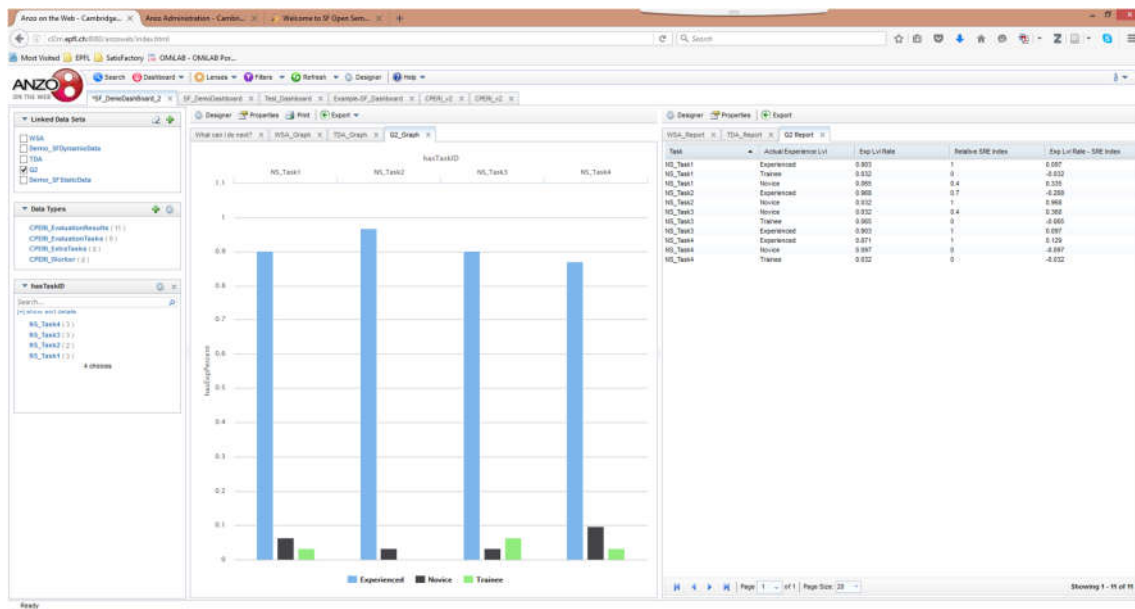


Figure 34 SRE Analysis – ANZO

Select Graph

N-RGA

LOAD GRAPH

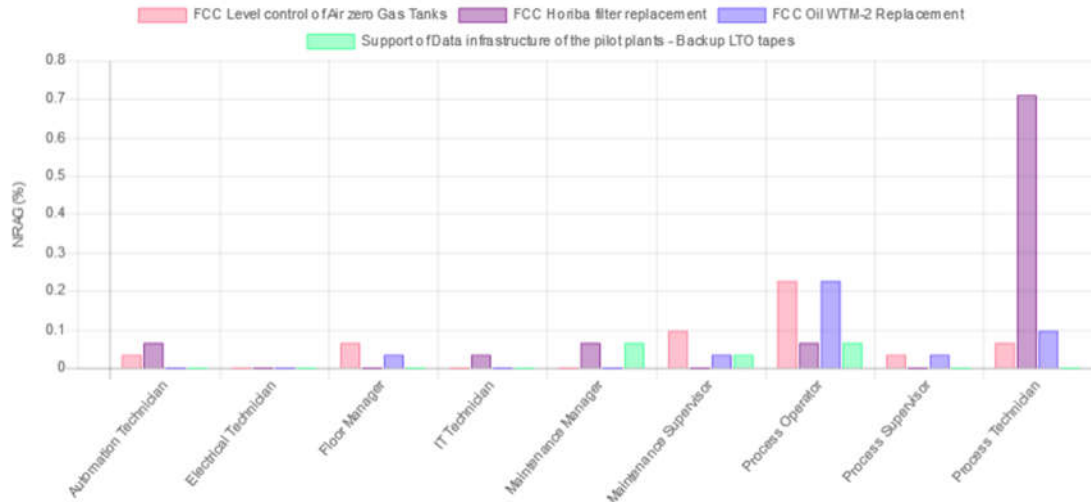


Figure 35 MWG Analysis – OSF

Select Graph

SRE

LOAD GRAPH

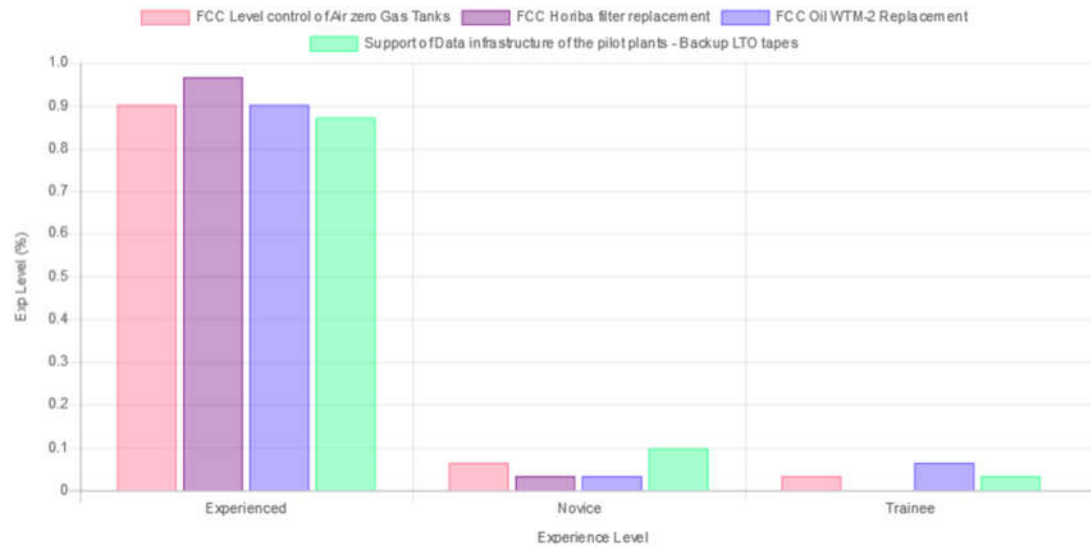


Figure 36 SRE Analysis - OSF

4.6 CONTEXT-AWARE ENGINE SUPPORT

The definition of some exploitation scenarios towards the support of specific SatisFactory components, such as the Decision Support System and the Context-Aware Engine, leads to investigate their potential interaction and consequent enhancement of the functionalities of the SatisFactory ecosystem. Section 4.5 introduced already a few pragmatic uses of the Semantic Framework to support the integrated Decision Support System (in short iDSS). The purpose of this section, instead, is to describe the role of the Semantic Framework, and in particular of the Ontology Manager, in the user interaction support for the Context-Aware Engine (see Figure 37Figure 37), which can gain from it in diverse ways that are described below.

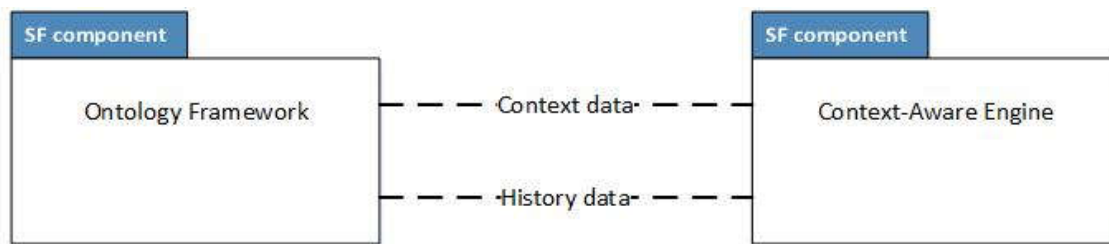


Figure 37 Context Aware Engine – OM interaction

Similarly to the approach used in the previous section, a few exploitation cases have been explored and then listed below:

- Analyze historical data to measure the effects of preventive procedures
- Video tagging to enrich the Context-Aware incident detection videos

The aforementioned scenarios have been explored in the framework of T2.4 and T3.3. In particular the Visualization toolkit for Re-adaption of existing facilities has a dedicated functionality for visualizing historical videos stored by the Context Aware Incident Detection module named Multiple Media Manager. Therefore, alongside the possibility for the supervisor to visualize historical videos of accidents occurred in the shop floor, information coming from the Semantic Framework are also provided to supervisors. Furthermore, thanks to adopted standard web technologies these functionalities can be used on heterogeneous devices.

The next two sections, therefore, will describe in details how the scenarios introduced above will be supported by the interaction of the Semantic Framework and the Context-aware engine, moreover, put the basis for the deployment of a Semantic Context Manager (SCM).

4.6.1 Analyze historical data to measure the effects of preventive procedures

The Ontology Manager can provide a valuable context for data analytics. In particular it might be used to analyze historical data, which are daily stored in the SatisFactory repository, and then support, for example, the assessment of the effects of preventive procedures.

In this context, the semantically-enriched framework can provide information about the performances of the workers, the occurrence of specific events, the monitored values/parameters regarding certain areas or machines in order to enable a subsequent assessment/evaluation process.

Therefore, according to Figure 39, the information to be taken into consideration are listed and further explained in Table 13. Unlike the analysis performed in Section 4.5.3, here the level of abstraction, which used to describe the information required to achieve an effective evaluation of the effects of the preventive procedures, is relatively higher.

Table 13 Semantic data to measure the effects of preventive procedures

Entity	Type of information	Notes
Historical data analysis	<ul style="list-style-type: none"> Measurement Events Maintenance Activities 	Beyond the so-called static datasets, the analysis of the effect of preventive procedures might require some detailed considerations about the maintenance activities that are scheduled and performed at the shop floor level. Moreover, the conjoint analysis of Measurement Events performed on specific areas/machines will further empower such assessment capabilities.
Static Data	<ul style="list-style-type: none"> Procedures Work Force Assets Work Areas 	

The implementation of the presented use case through the use of ANZO Enterprise allows a friendly visualization of the information that should be analysed from the end user to set evaluation criteria and custom thresholds on the results values, exploiting the networked and semantics-enriched nature of the data (see Figure 38).

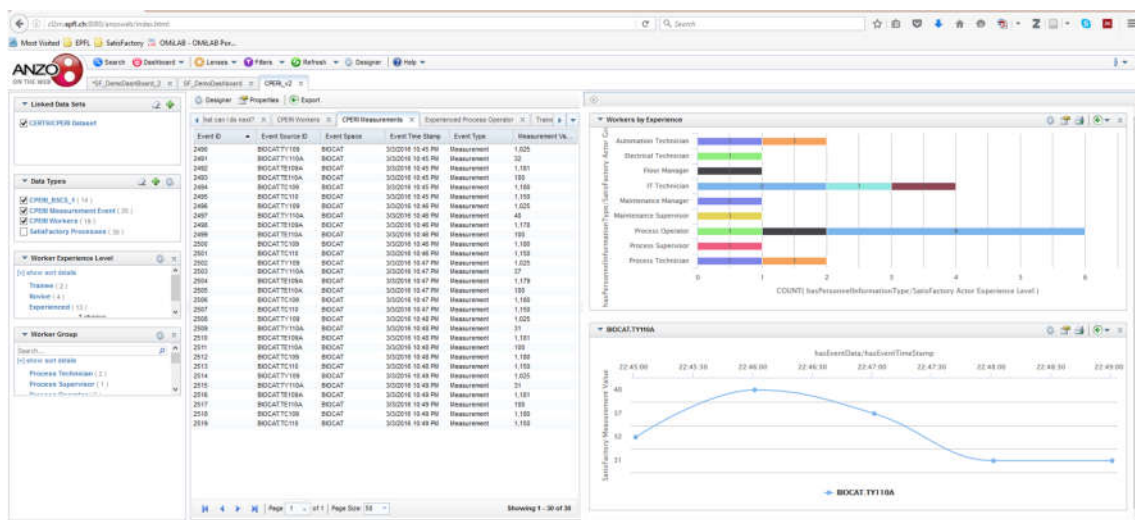


Figure 38 ANZO Dashboard example: Filtered Events, Involved Workforce, Measurements

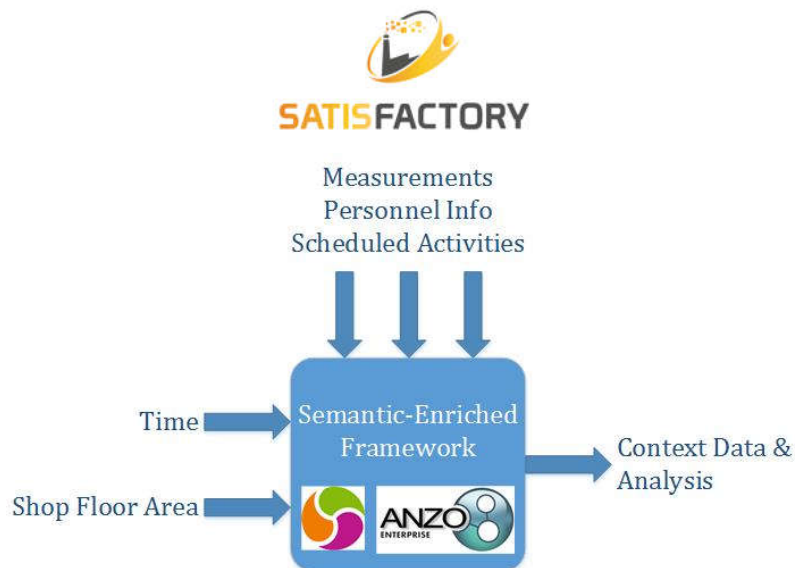


Figure 39 Black Box: Historical Data Analysis

The Visual toolkit for Re-adaptation of existing facilities has the aim to leverage on information coming from the Semantic Context Manager. A placeholder in the GUI structure has been already integrated at the current stage of development a subset of contextual information are represented.

The picture below (Figure 40) represents a screenshot of a usage of the tool in the pre-pilot activity in CPERI.

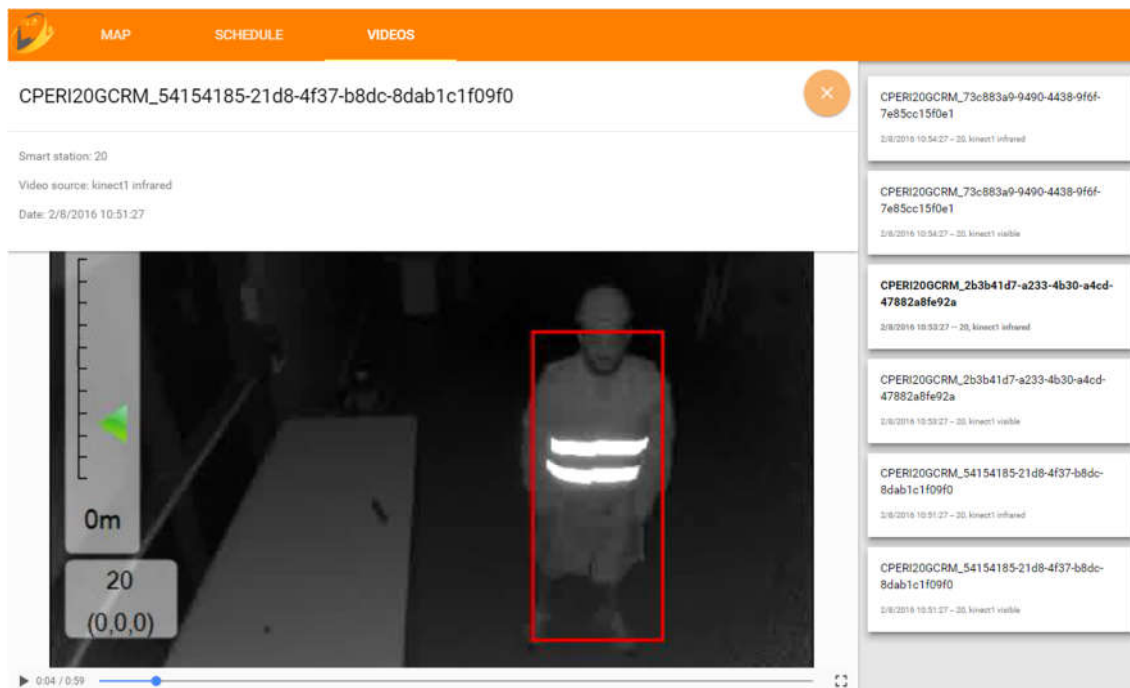


Figure 40 Context Aware information added to an incident video

Noticeably, the video has a drop down menu where the information are embedded. For the moment only high level information are displayed such as:

- the ID of the Smart Assembly Station in which the accident has occurred

- the camera model and the video source (in this case is a privacy preserving infrared camera)
- the precise date and time where the video has been recorded

In the next implementation phases, following the indications above (Table 13) more enriched information will be added and the Semantic Context Manager will, then, elaborate more complex analysis in order to detail with a finer level the information describing the incident.

4.6.2 Video tagging to enrich the Context-Aware incident detection videos

The requirements specification document already documented in D2.2 introduced the need of defining a common base for the interaction between the Context-aware incident detection engine (T3.3) and the Ontology Manager. The objective of task T3.3 is to develop a context manager module which supports workers' safety and comfort by leveraging location information as well as media and gesture recognition data. The context manager will include enabling modules such as the Localization Manager, the Multiple-Media Manager and the Gesture & Content Recognition Manager. In particular, the purpose of the Multiple-Media Manager is to automatically process multimedia in order to enrich localization data with visual information and to support the detection of incidents.

In this framework, the introduction of a video annotation vocabulary together with the provision of enriched data to be used on incident videos to enrich their description represent a relevant added value for this project (see Figure 33).

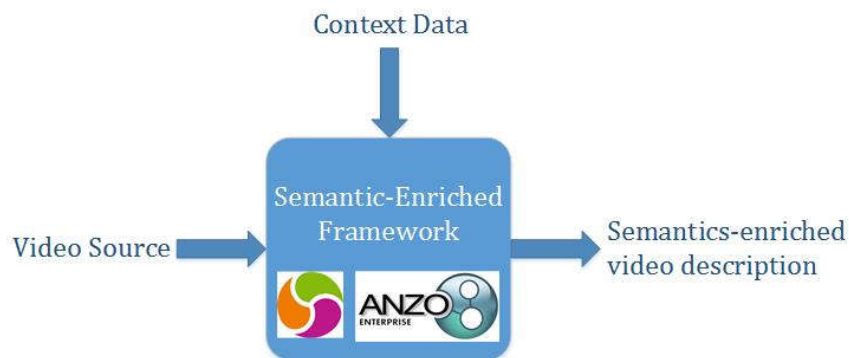


Figure 41 Black Box: Semantics-Enrichment of videos

Regarding the video metadata, the Multi-Media Manager transports them using a combination of EXIF format for MJPEG and video overlay. EXIF allows insertion of companion information to JPEG images in form of key - value. Moreover, EXIF defines both the keys (tags) and values format (a complete list of tags is available at the following link <http://www.exiv2.org/tags.html>).

Therefore, starting from the available information and format, we aim to enrich the video description in a way that the latter can store a complete set of information that might be used to properly assess the context of the related event.

An example of semantics-enriched incident video annotation is presented below:

```

PREFIX sf: <http://ict4sm.epfl.ch/.../sf#>
<http://example.com/video> a sf:Video ;
    dc:title "Incident1234" ;
    sf:videoType: IncidentVideo;
  
```

```
sf:recordedBy "Camera1234" ;
sf:LocatedIn "Area1234" ;
sf:videoLenght "13" ;
sf:EventTime "10.58AM" ;
sf:EventDate "24/06/2016" ;
sf:WorkerInThisarea Worker1 ;
sf:WorkerInThisarea Worker2 ;
sf:ActivityInThisArea Task42 .
```

This mechanism certainly requires a strong integration between the CIDEM and the Ontology Manager in order to retrieve the necessary information on triggered events (e.g., request from the Context Aware Engine), and promptly RDFization of this, which will be then attached to the video (see section 4.4.2). Therefore, enriched information regarding the incident events might be extracted through POST calls from the Semantic framework repository as already described in the previous section. Table 14 summarizes the type of information that might be exchanged within the Semantic Context Manager, which will support the semantics-enrichment of the incident video.

Table 14 Semantic annotations for incident videos

Entity	Type of information	Notes
Historical Data	<ul style="list-style-type: none"> Measurements Events Incident Events Shop floor activities Alert Events Area restrictions 	
Static Data	<ul style="list-style-type: none"> Shop floor activities Work Force Assets Work Areas 	

These different types of information will, therefore, allow supervisors to further investigate historical videos of accident coming from the shop floor level. In the following picture is provided an example of how these information are embedded in the video and visualized.

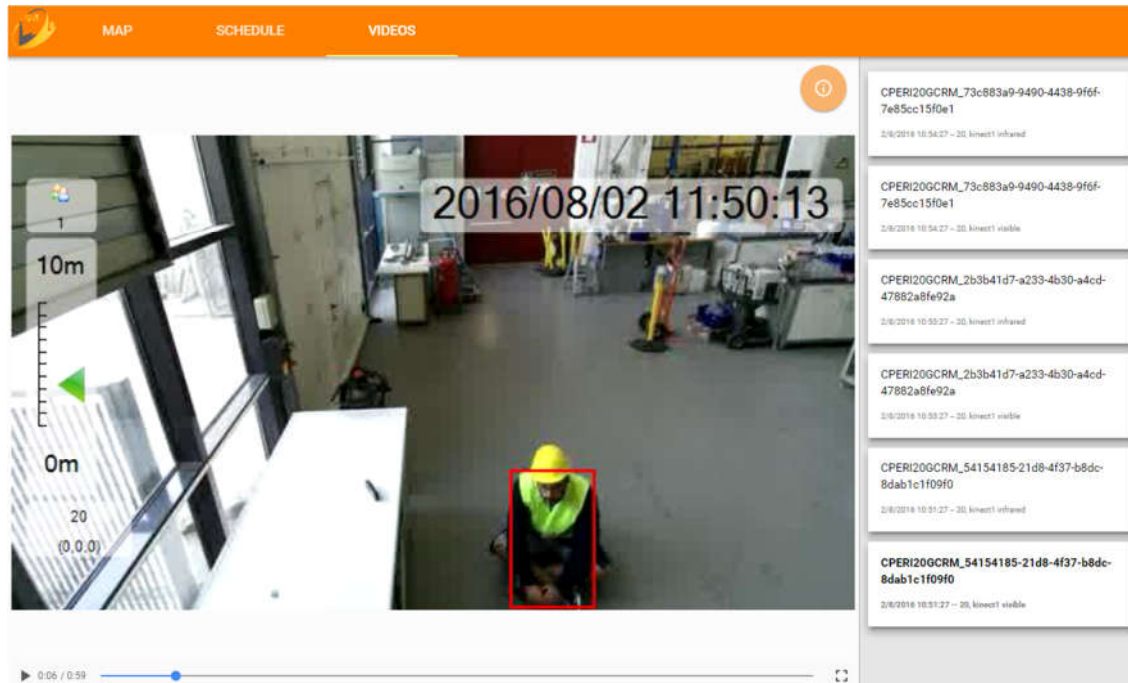


Figure 42 - Video enrichment example

In specific, the following information are added while encoding the video in case of accident:

- people count
- distance of the area from the camera in which the accident has occurred
- Smart Assembly Station ID
- Smart Assembly Station coordinates based on Localization Manager axis and references
- date and time of the accident
- bounding box surrounding the involved personnel

These information can help supervisors to determine potential risks factor during the incident and could start a behaviour change procedure in order to prevent the same accident to happen again.

In the next implementation phases, following the indications above (Table 14) more enriched information will be added in order to perform more complex analysis and enhance the analytics capacities through semantic annotations.



5. CONCLUSIONS

In this deliverable regarding the development of a *Semantically-enriched framework for analysis and design of dynamically evolving shop floor operations* we presented the implementation phase of the semantic structures and knowledge engineering methodologies documented in D2.2. The networked architecture of the Ontology Manager has been designed and then deployed with the use of state-of-the-art tools for knowledge modelling and semantic interoperability.

The analysis of the dynamic shop floor operations is presented and supported by leveraging the semantics-enriched nature of the datasets flowing through the Ontology Manager, and then provided to the downstream components and end users as a valuable basis for further detailed investigations of the SatisFactory shop floor information.

Some application scenarios are, therefore, presented aiming to describe the interaction of the Ontology Manager with two other core SatisFactory components, such as the integrated Decision Support System and the Context-Aware engine. These show how the semantics-enriched information flow can be pragmatically exploited for analysis and design of dynamically evolving shop floor operations. More details regarding their implementation and use will be provided in the framework of T3.3 and T3.5 which will be finalized by the end of M30.

In conclusion, this deliverable documents the effort spent between the third and twentieth month of the project and represents the final status of the activities performed in Task 3.1. The models presented in this document and the application cases tested by data, which have been gathered at the CPERI shop floor, should be perceived as a living asset that will grow while SatisFactory project proceeds.

REFERENCES

1. *Anzo Enterprise*. Cambridge Semantics, 2014.
2. Matthew Horridge, Sean Bechhofer, 2009. "The OWL API: A Java API for Working with OWL 2 Ontologies," *OWLED 2009, 6th OWL Experienced and Directions Workshop*, Chantilly, Virginia, October 2009
3. "Anzo Smart Data Platform | Cambridge Semantics." [Online]. Available: <http://www.cambridgesemantics.com/technology/anzo-smart-data-platform>. [Accessed: 19-May-2015].
4. "Anzo Smart Data Architecture | Cambridge Semantics." [Online]. Available: <http://www.cambridgesemantics.com/technology/anzo-smart-data-architecture>. [Accessed: 19-May-2015].
5. "Anzo Smart Data Access | Cambridge Semantics." [Online]. Available: <http://www.cambridgesemantics.com/technology/anzo-smart-data-access>. [Accessed: 19-May-2015].
6. Abdul-Ghafour, S., Ghodous, P. and Shariat, B., 2012, August. Integration of product models by ontology development. In *Information Reuse and Integration (IRI), 2012 IEEE 13th International Conference on* (pp. 548-555). IEEE.
7. Abdul-Ghafour, S., Ghodous, P., Shariat, B. and Perna, E., 2011. Ontology Development for the Integration of CAD Models in a Collaborative Environment. In *Improving Complex Systems Today* (pp. 207-214). Springer London.
8. Dey, A.K., 2001. Understanding and using context. *Personal and ubiquitous computing*, 5(1), pp.4-7.
9. "LinkedDesign." [Online]. Available: <http://www.linkeddesign.eu/>. [Accessed: 09-Apr-2015].
10. P. Cimiano, A. Hotho, and S. Staab. Learning concept hierarchies from text corpora using formal concept analysis. *Journal of Artificial Intelligence Research*, 24:305-339, Aug. 2005.
11. Unbehauen, Jörg, et al. "Knowledge extraction from structured sources." *Search Computing*. Springer Berlin Heidelberg, 2012. 34-52.
12. SatisFactory project deliverables. D1.2 Use Case Analysis and application scenarios description
13. SatisFactory project deliverables. D2.2 Knowledge modelling to support the human resource optimization
14. SatisFactory project deliverables. D3.5 Shop floor feedback engine and DSS.

ANNEX

ANNEX A

Table 15 Detailed description of the worker groups

Group Name	Set of Skills
Floor Manager	<ul style="list-style-type: none"> • decision making and decision support for allocation of resources at existing processes • overall supervision on the workflow of the shop floor • good knowledge on chemical processes • management of human resources • administrative skills • able to organize and inspect the general workflow of the shop floor • collaboration skills with other stakeholders • continuous optimization of the workflow of the shop floor
Process Supervisor	<ul style="list-style-type: none"> • schedule new constructions or system revamps • share of information about daily experiments and processing of data from the field and the laboratory analysis • experience on chemical processes • excellent knowledge of Pilot Plants operation • good knowledge of using process monitoring systems • administrative skills • responsible for the operation of the pilot plants • manage process operators
Maintenance Manager	<ul style="list-style-type: none"> • sharing of programmed actions • be informed about existing work in progress and involvement of technical team • schedule new constructions or system revamps • able to organize tasks on the shop floor • collaboration skills with other stakeholders • automation, electrical and mechanical knowledge • keep the general view of the shop floor on high levels • able to have good collaboration with other stakeholders
Maintenance Supervisor	<ul style="list-style-type: none"> • sharing of programmed actions • be informed about existing work in progress and involvement of technical team • able to coordinate the technical users • collaboration skills with other stakeholders and user groups • automation, electrical and mechanical knowledge • keep the general view of the shop floor on high levels • able to have good collaboration with other stakeholders

	<ul style="list-style-type: none"> • knowledge about all technical issues about the shop floor and the operation of the pilot plants • able to respond fast and efficient to incidents and unplanned situations
Process Technician	<ul style="list-style-type: none"> • Preserve information related to performed procedures • Support and maintain Pilot plants operation • Involved in new constructions and revamps • good knowledge of mechanical design software such as AutoCAD Mechanical • able to use tools for construction of Pilot Plants such as electric winding, electric drills and all electric tools • able to respond fast and efficient to incidents and unplanned situations • able to handle immediate and crucial malfunctions, able to organize scheduled actions in order not to interrupt the normal operation of the shop floor • handles process supervisors' and process operators' queries on not scheduled tasks, cooperate with maintenance managers and supervisors to schedule the maintenance tasks
Control, Automation, IT Technician	<ul style="list-style-type: none"> • Design information (P&ID, mechanical drawings) • Preserve information related to performed procedures • Maintaining IT structures • Involved in new constructions and revamps • experience on SCADA and automation software • very good knowledge on IT infrastructures, Windows based programs • able to communicate and cooperate with other user groups • able to respond fast and efficient to incidents and unplanned situations • able to handle immediate and crucial malfunctions • able to organize scheduled actions in order not to interrupt the normal operation of the shop floor • handles process supervisors' and process operators' queries on not scheduled tasks • cooperate with maintenance managers and supervisors to schedule the maintenance tasks
Electrical Technician	<ul style="list-style-type: none"> • design information (P&ID, electrical schematics) • preserve information related to performed procedures • support and maintain Pilot plants operation • involved in new constructions and revamps • good knowledge and experience on high and low voltage electrical systems • able to understand electrical schematics and manuals • good knowledge of electrical design software such as AutoCAD Electrical • able to respond fast and efficient to incidents and unplanned situations • able to handle immediate and crucial malfunctions

	<ul style="list-style-type: none"> • handles process supervisors' and process operators' queries on not scheduled tasks • cooperate with maintenance managers and supervisors to schedule the maintenance tasks
Process Operator	<ul style="list-style-type: none"> • Perform Daily operations • Report about malfunctions or abnormal operation of equipment or systems • Organizing raw materials for the experiments on the Pilot Plants • experience on chemical processes • usage of hydraulic and pneumatic systems • good knowledge of safety measures on Pilot Plants operation • good knowledge of using process monitoring systems • able to operate the pilot plants with efficiency and fast response to incidents and unplanned situations • able to recognize malfunctions and abnormal situations regarding the Pilot Plants

ANNEX B

Event Work Schedule XSL code example

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
    xmlns:xsl =
        "http://www.w3.org/1999/XSL/Transform"
    xmlns:xsd = "http://www.w3.org/2001/XMLSchema"
    xmlns:b2mml = "http://www.mesa.org/xml/B2MML-
V0600"
    xmlns:rdf = "http://www.w3.org/1999/02/22-
rdf-syntax-ns#"
    xmlns:rdfs = "http://www.w3.org/2000/01/rdf-
schema#"
    xmlns:sf =
        "http://ict4sm.epfl.ch/files/content/sites/licp/files/ontologie
s/sf#"
    xmlns:iron =
        "http://purl.org/ontology/iron#">

    <xsl:output method="xml" indent="yes"/>

    <xsl:template match="/">
        <rdf:RDF>
            <xsl:apply-templates/>
        </rdf:RDF>
    </xsl:template>

    <xsl:template match="ShopFloor">
        <xsl:apply-templates select="SFEvents"/>
    </xsl:template>

    <xsl:template match="SFEvents">
        <xsl:apply-templates select="EventList"/>
    </xsl:template>

    <xsl:template match="EventList">
        <xsl:apply-templates select="Event"/>
    </xsl:template>

    <xsl:template match="Event">
        <xsl:variable name="SatisfactoryEventID"><xsl:value-of
select="ID"/></xsl:variable>
        <rdf:Description
rdf:about="http://ict4sm.epfl.ch/files/content/sites/licp/files/onto
logies/datasets/ET_{ $SatisfactoryEventID }">
            <rdf:type
rdf:resource="http://ict4sm.epfl.ch/files/content/sites/licp/files/o
ntologies/sf#CPERI_EvaluationTasks"/>
```

```

        <xsl:apply-templates select="Content"/>
    </rdf:Description>
</xsl:template>

<xsl:template match="Content">
    <xsl:apply-templates select="Maintenance"/>
</xsl:template>

<xsl:template match="Maintenance">
    <xsl:apply-templates select="WorkScheduleList"/>
</xsl:template>

<xsl:template match="WorkScheduleList">
    <xsl:apply-templates select="b2mml:WorkRequest"/>
</xsl:template>

<xsl:template match="b2mml:WorkRequest">
    <xsl:apply-templates select="b2mml:JobOrder"/>
</xsl:template>

    <xsl:template match="b2mml:JobOrder">
        <xsl:variable name="StartDate"><xsl:value-of
select="substring(b2mml:StartTime,0,11)"/></xsl:variable>
        <xsl:variable name="StartTime"><xsl:value-of
select="substring(b2mml:StartTime,12,8)"/></xsl:variable>
        <xsl:variable name="EndDate"><xsl:value-of
select="substring(b2mml:EndTime,0,11)"/></xsl:variable>
        <xsl:variable name="EndTime"><xsl:value-of
select="substring(b2mml:EndTime,12,8)"/></xsl:variable>
        <sf:hasDuration
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"><xsl:value-of
select="days-from-duration(xsd:date($EndDate) -
xsd:date($StartDate))*24 + hours-from-duration(xsd:time($EndTime) -
xsd:time($StartTime)) + (minutes-from-duration(xsd:time($EndTime) -
xsd:time($StartTime)) div 60) + (seconds-from-
duration(xsd:time($EndTime) - xsd:time($StartTime)) div
3600)"/></sf:hasDuration>
        <sf:hasTaskID
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">NS_Task<xsl:v
alue-of select="b2mml:ID"/></sf:hasTaskID>
        <xsl:apply-templates
select="b2mml:PersonnelRequirement"/>
    </xsl:template>

    <xsl:template match="b2mml:PersonnelRequirement">
        <sf:hasActorID
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"><xsl:value-of
select="b2mml:PersonID"/></sf:hasActorID>
    </xsl:template>

</xsl:stylesheet>

```

ANNEX C

WSA POST call

http://160.40.51.99:8890/sparql/?default-graph-uri=&query=PREFIX xsd%3A+%3Chttp%3A%2F%2Fwww.w3.org%2F2001%2FXMLSchema%23%3E%0D%0APREFIX%3A+%3Chttp%3A%2F%2Fict4sm.epfl.ch%2Ffiles%2Fcontent%2Fsites%2Flicp%2Ffiles%2Ffontologies%2Fsf%23%3E%0D%0ASELECT+%3FTaskID+%3FRAG+%3FREL+%0D%0A%23%3FSRE_E+%3FSRE_N+%3FSRE_T+%0D%0A%23%3FMWG_FM+%3FMWG_PS+%3FMWG_MM+%3FMWG_MS+%3FMWG_PO+%3FMWG_PT+%3FMWG_ET+%3FMWG_AT+%3FMWG_IT%0D%0A%3FACTORID+%3FGroup+%3FExLvl+%0D%0A%23%3FSRE+%3FMWG%0D%0A%28xsd%3Adouble%28%3FMWG%29*xsd%3Adouble%28%3FSRE%29+AS+%3FSuitability%29%0D%0AWHERE+{%0D%0A%3FTask+a+sf%3ACPERI_ExtraTasks+.+%3FTask+sf%3AhasTaskID+%3FTaskID+.+%3FTask+sf%3AhasRequiredActorGroup+%3FRAG+.+%3FTask+sf%3AhasRequiredExperienceLevel+%3FREL+.+%0D%0A%3FTask+sf%3AhasSREIndexWith_ Experienced+%3FSRE_E+.+%3FTask+sf%3AhasSREIndexWith_Novice+%3FSRE_N+.+%3FTask+sf%3AhasSREIndexWith_Trainee+%3FSRE_T+. %0D%0A%0D%0A%3FRAG+sf%3AhasMWGIndexWith_FloorManager+%3FMWG_FM+.+%3FRAG+sf%3AhasMWGIndexWith_ProcessSupervisor+%3FMWG_PS+.+%3FRAG+sf%3AhasMWGIndexWith_MaintenanceManager+%3FMWG_MM+.+%09%0D%0A%3FRAG+sf%3AhasMWGIndexWith_MaintenanceSupervisor+%3FMWG_MS+.+%3FRAG+sf%3AhasMWGIndexWith_ProcessOperator+%3FMWG_PO+.+%3FRAG+sf%3AhasMWGIndexWith_ProcessTechnician+%3FMWG_PT+.+%0D%0A%3FRAG+sf%3AhasMWGIndexWith_ElectricalTechnician+%3FMWG_ET+.+%3FRAG+sf%3AhasMWGIndexWith_AutomationTechnician+%3FMWG_AT+.+%3FRAG+sf%3AhasMWGIndexWith_ITTechnician+%3FMWG_IT+. %0D%0A%0D%0A%3FACTOR+a+sf%3ACPERI_Worker+.+%3FACTOR+sf%3AhasActorID+%3FACTORID+.+%3FGroup+a+sf%3AACTORGroup+.+%3FACTOR+sf%3AhasActorGroup+%3FGroup+.+%3FExLvl+a+sf%3AExperienceLevel+.+%3FACTOR+sf%3AhasExperienceLevel+%3FExLvl+. +%0D%0A%0D%0A%0D%0ABIND%28+IF%28%3FExLvl%3D+sf%3AExperienced%2C+%3FSRE_E%2C++IF%28%3FExLvl%3D+sf%3ANovice%2C+%3FSRE_N%2C++IF%28%3FExLvl%3D+sf%3ATrainee%2C+%3FSRE_T%2C+%22%22%29%29%29+AS+%3FSRE%29.%0D%0A%0D%0ABIND%28+IF%28%3FGroup%3D+sf%3AFloorManager%2C+%3FMWG_FM%2C+IF%28%3FGroup%3D+sf%3AProcessSupervisor%2C+%3FMWG_PS%2CIF%28%3FGroup%3D+sf%3AMaintenanceManager%2C+%3FMWG_MM%2CIF%28%3FGroup%3D+sf%3AMaintenanceSupervisor%2C+%3FMWG_MS%2CIF%28%3FGroup%3D+sf%3AProcessOperator%2C+%3FMWG_PO%2CIF%28%3FGroup%3D+sf%3AProcessTechnician%2C+%3FMWG_PT%2CIF%28%3FGroup%3D+sf%3AElectricalTechnician%2C+%3FMWG_ET%2CIF%28%3FGroup%3D+sf%3AAutomationTechnician%2C+%3FMWG_AT%2CIF%28%3FGroup%3D+sf%3AITTechnician%2C+%3FMWG_IT%2C%22%22%29%29%29%29%29%29%29%29%29+AS+%3FMWG%29+. %0D%0A%0D%0AFILTER+%28%3FTaskID%3D%22NS_Task1%22^^xsd%3Astring+| |+%3FTaskID%3D%22NS_Task2%22^^xsd%3Astring+| |+%3FTaskID%3D%22NS_Task3%22^^xsd%3Astring+| |+%3FTaskID%3D%22NS_Task4%22^^xsd%3Astring%29%0D%0A}%0D%0AORDER+BY+%28%3FTaskID%29&should-sponge=&format=application%2Frd%2Bxml&timeout=0&debug=on

SPARQL Query (WSA) RDF/XML outcome example

```
<rdf:RDF>
  <rdf:Description rdf:nodeID="rset">
    <rdf:type rdf:resource="http://www.w3.org/2005/sparql-results#ResultSet"/>
    <res:resultVariable>TaskID</res:resultVariable>
    <res:resultVariable>RAG</res:resultVariable>
    <res:resultVariable>REL</res:resultVariable>
    <res:resultVariable>ActorID</res:resultVariable>
    <res:resultVariable>Group</res:resultVariable>
    <res:resultVariable>ExLvl</res:resultVariable>
    <res:resultVariable>Suitability</res:resultVariable>
    <res:solution rdf:nodeID="r0">
      <res:binding rdf:nodeID="r0c0">
        <res:variable>TaskID</res:variable>
        <res:value datatype="http://www.w3.org/2001/XMLSchema#string">NS_Task1</res:value>
      </res:binding>
      <res:binding rdf:nodeID="r0c1">
        <res:variable>RAG</res:variable>
        <res:value rdf:resource="http://ict4sm.epfl.ch/.../sf#ElectricalTechnician"/>
      </res:binding>
      <res:binding rdf:nodeID="r0c2">
        <res:variable>REL</res:variable>
        <res:value rdf:resource="http://ict4sm.epfl.ch/.../sf#Experienced"/>
      </res:binding>
      <res:binding rdf:nodeID="r0c3">
        <res:variable>ActorID</res:variable>
        <res:value datatype="http://www.w3.org/2001/XMLSchema#int">8</res:value>
      </res:binding>
      <res:binding rdf:nodeID="r0c4">
        <res:variable>Group</res:variable>
        <res:value rdf:resource="http://ict4sm.epfl.ch/.../sf#ProcessOperator"/>
      </res:binding>
      <res:binding rdf:nodeID="r0c5">
        <res:variable>ExLvl</res:variable>
        <res:value rdf:resource="http://ict4sm.epfl.ch/.../sf#Experienced"/>
      </res:binding>
      <res:binding rdf:nodeID="r0c6"><res:variable>Suitability</res:variable>
        <res:value datatype="http://www.w3.org/2001/XMLSchema#double">0.1</res:value>
      </res:binding>
    </res:solution>
    ...
    ...
    ...
  </rdf:Description>
</rdf:RDF>
```



SPARQL Query (MGW Analysis) RDF/XML outcome example

```
<rdf:RDF>
  <rdf:Description rdf:nodeID="rset">
    <rdf:type rdf:resource="http://www.w3.org/2005/sparql-results#ResultSet"/>
    <res:resultVariable>TaskID</res:resultVariable>
    <res:resultVariable>Group</res:resultVariable>
    <res:resultVariable>RAG</res:resultVariable>
    <res:resultVariable>Count</res:resultVariable>
    <res:resultVariable>tot</res:resultVariable>
    <res:resultVariable>NonRAG_rate</res:resultVariable>
    <res:resultVariable>MWG</res:resultVariable>
    <res:resultVariable>Diff_MWG_index</res:resultVariable>
    <res:solution rdf:nodeID="r0">
      <res:binding rdf:nodeID="r0c0">
        <res:variable>TaskID</res:variable>
        <res:value datatype="http://www.w3.org/2001/XMLSchema#string">NS_Task1</res:value>
      </res:binding>
      <res:binding rdf:nodeID="r0c1">
        <res:variable>Group</res:variable>
        <res:value rdf:resource="http://ict4sm.epfl.ch/.../sf#ProcessOperator"/>
      </res:binding>
      <res:binding rdf:nodeID="r0c2">
        <res:variable>RAG</res:variable>
        <res:value rdf:resource="http://ict4sm.epfl.ch/.../sf#ElectricalTechnician"/>
      </res:binding>
      <res:binding rdf:nodeID="r0c3">
        <res:variable>Count</res:variable>
        <res:value datatype="http://www.w3.org/2001/XMLSchema#integer">7</res:value>
      </res:binding>
      <res:binding rdf:nodeID="r0c4">
        <res:variable>tot</res:variable>
        <res:value datatype="http://www.w3.org/2001/XMLSchema#integer">31</res:value>
      </res:binding>
      <res:binding rdf:nodeID="r0c5"><res:variable>NonRAG_rate</res:variable>
        <res:value datatype="http://www.w3.org/2001/XMLSchema#double">0.225806</res:value>
      </res:binding>
      <res:binding rdf:nodeID="r0c6">
        <res:variable>MWG</res:variable>
        <res:value>0.1</res:value>
      </res:binding>
      <res:binding rdf:nodeID="r0c7">
        <res:variable>Diff_MWG_index</res:variable>
        <res:value datatype="http://www.w3.org/2001/XMLSchema#double">-0.125806</res:value>
      </res:binding>
    </res:solution>
    ...
    ...
    ...
  </rdf:Description>
</rdf:RDF>
```