Proceedings of DET2011
7th International Conference on Digital Enterprise Technology
Athens, Greece
28-30 September 2011

VIRTUAL FACTORY MANAGER OF SEMANTIC DATA

Giorgio Ghielmini
ICIMSI-SUPSI
giorgio.ghielmini@supsi.ch

Paolo Pedrazzoli
ICIMSI-SUPSI
paolo.pedrazzoli@supsi.ch

Diego Rovere
ICIMSI-SUPSI
diego.rovere@supsi.ch

Walter Terkaj
ITIA-CNR
walter.terkaj@itia.cnr.it

Claudio R. Boër
ICIMSI-SUPSI
claudio.boer@supsi.ch

Giovanni Dal Maso
Technology Transfer System S.r.l.
dalmaso@ttsnetwork.com

Ferdinando Milella
SimX ltd.
f.milella@simx.co.uk

Marco Sacco
ITIA-CNR
marco.sacco@itia.cnr.it

ABSTRACT

The growing importance of manufacturing SMEs within the European economy, in terms of Gross Domestic Product and number of jobs, emphasizes the need of proper ICT tools to support their competitiveness. Major ICT players already offer one-does-all Product Lifecycle Management suites, supporting several phases of the product-process-plant definition and management. However, these do also show consistent shortcomings in terms of SME accessibility, degree of personalization and they often lack of an acceptable level of interoperability. These problems are being addressed by the development of a Virtual Factory Framework (VFF), within an EU funded project. The approach is based on four pillars: 1) Semantic Shared Data Model, 2) Virtual Factory Manager (VFM), 3) Decoupled Software Tools that lay on the shared data model and can interact through the VFM, 4) Integration of Knowledge. This paper will focus on the Virtual Factory Manager, proposing an evolution of the former VFF second Pillar (Sacco et al, 2010), that acts as a server supporting the I/O communications within the framework and its stored knowledge for the decoupled software tools needing to access its repository.

KEYWORDS

Virtual Factory, Enterprise Modelling, Reference Model, Interoperability, Semantic Data Model

1. INTRODUCTION

Market needs and expectations require a continuously rapidly evolving production framework: thus production systems, from small to large scale and integrated factories, have to be conceived and set-up in shorter and shorter times (Chryssolouris et al, 2008). Several critical aspects, related to this need of rapid prototyping of factories, have to be addressed: it is critical to provide sufficient product variety to meet customer requirements, business needs and technical advancements (Huang et al, 2005), while maintaining economies of scale and scope within the manufacturing processes (Terkaj et al, 2009). Therefore, the current challenge in manufacturing engineering consists in the innovative integration of the product, process and factory worlds and the related data, aiming at synchronizing their lifecycles (Tolio et al, 2010).

The creation of a holistic, integrable, up-gradable, scalable Virtual representation of the Factory can empower this synchronization, promoting high cost savings in the implementation of new manufacturing facilities or reconfiguration

of existing ones, thanks to the effective virtual representation of buildings, resources, process, and products: this is shown both by industrial practice and academic scientific research. The entire factory is simulated as a continuous and consistent digital model, which can be used, without interruption, all the way from the product idea to the final dismantling of the production plants and buildings (Bracht and Masurat, 2005).

These challenge is being addressed by the development of a Virtual Factory Framework (VFF), within an EU funded project (http://www.vff-project.eu/). The approach is based on four pillars: 1) Semantic Shared Virtual Factory Data Model (VFDM), 2) Virtual Factory Manager (VFM), 3) Decoupled Software Tools, based on the VFDM and that can interact through the VFM, and 4) Integration of Knowledge. VFF objective is to fosters an integrated virtual environment that supports factory processes along all the phases of its lifecycle.

This paper will focus on the Virtual Factory Manager (VFM), proposing an evolution of the former VFF Pillar II (Sacco et al, 2010). This evolution finds its justification in the identified weakness of the former second Pillar, found both in the support to the data consistency check against modifications performed by different modules and in the integration of the knowledge layer with the

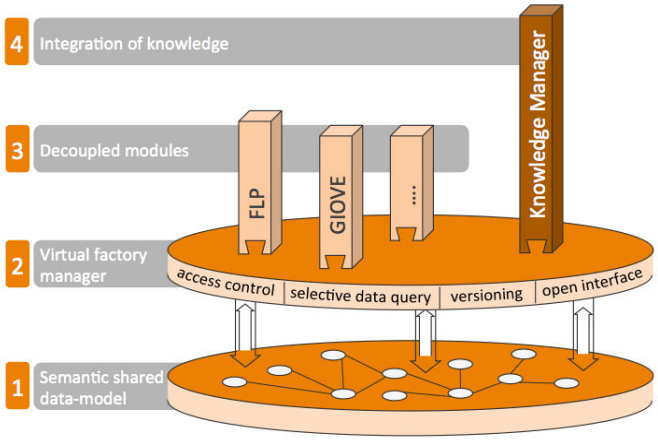

Figure 1 - The Semantic Virtual Factory Framework architecture

pure factory data layer. A viable solution has been identified in the adoption of ontology as means for data and relationships representation, promoting knowledge integration in the data-model. This approach introduces a modification in the overall VFF architecture, where pillar IV (knowledge integration) is no longer seen as foundation of Pillar I (reference data model), as presented in (Sacco et al, 2010), but rather considered as an additional decoupled module (Figure 1).

This paper presents the new VFF framework born from this evolution.

2. VIRTUAL FACTORY FRAMEWORK

As mentioned, an answer to the market requirements previously highlighted has been provided by the development of a first version of the Virtual Factory Manager (Sacco et al, 2011). That solution proved the validity of the concept of having an integrated virtual environment supporting the design and management of all the factory entities, ranging from the single product to the network of companies, along all the stages of the factory lifecycle. The centralized data management platform based on a common description of the digital factory demonstrated the capability to improve the integration process between software design tools (existing and new developed ones) and to provide a shared knowledge base to be used during the factory modelling phases. Nevertheless, the former approach showed some weaknesses both in the support to the data consistency check against modifications to parts of the factory instances performed by different modules and in the integration of the knowledge layer with the pure factory data layer. The result, highlighted by advanced tests, was a framework affected by some problems of usability.

The main cause of this situation has been identified in the impossibility of the previous implementation of the VFDM, to represent not only valid data structures, but also their semantics. A viable solution has been identified in the adoption of ontology as means for data and relationships representation in order to improve the integration of knowledge among the VFF pillars. This approach introduces some modifications in the whole VFF picture, affecting the way the components of the architecture interact. Therefore, the result is a tighter cooperation between the Knowledge Manager and the VFDM pillars.

2.1. SEMANTIC VIRTUAL FACTORY FRAMEWORK ARCHITECTURE

Figure 1 shows the new architecture of the Semantic Virtual Factory Framework composed by the four pillars of Semantic Shared Data Model (Pillar I), Semantic VF Manager (Pillar II), Decoupled VF Modules (Pillar III) and Knowledge Manager (Pillar IV).

The Semantic VFDM establishes a coherent standard extensible set of ontologies for the common integrated representation of the factory objects and of the factory knowledge domain, basing on the tools of the semantic web (mainly the Web Ontology Language - OWL). Section 3 is dedicated to a thorough description of the new

Semantic VFDM approach and of the reasons that drove the change.

This common ontology set is governed by the Semantic VFM (Pillar II) that completes the functionalities of access control, data versioning and selective data query, already implemented by the previous VFM, with a full support to the semantic data validation. In this way, Decoupled Modules (Pillar III) modifying single parts of the factory data immediately receive feedback on the consistency of their actions with the overall definition of the factory instance. Section 4 and Section 6 are respectively dedicated to the analysis of the Semantic VFM and to the description of the current prototype implementation.

In order to couple with the new features of the VFM and with the new data exchange formats, it has been necessary to intervene on the internal architecture of the Decoupled Modules and, in particular, of their VF Connector modules. Section 5 reports on the new structure of Pillar III, while an example of new module interacting with the Semantic VFM prototype is provided in Section 7. Also the Knowledge Manager (Pillar IV), that was the only component of the previous architecture already based on the usage of ontologies, has been affected by the new approach. With the new structure, in fact, it can directly interface the Semantic VFM as a decoupled module managing a dedicated part of the Semantic Data Model ontology. In this way it has been removed another weak point of VFF that was represented by the need for adaptation of formats and protocols between pillars.

3. SHARED SEMANTIC DATA MODEL

The Reference Model (Pillar I) establishes a coherent standard extensible Virtual Factory Data Model (VFDM) for the common representation of factory objects related to production systems, resources, processes and products. The common data model can be considered as the shared meta-language providing a common definition of the data that will be governed by the VFM (Pillar II) and used and updated by the Decoupled Functional Modules (Pillar III).

According to the original requirements, the VFDM has to be holistic, covering all the relevant fields related to the Factory domain and exploit existing technical standards to represent the data. Moreover the VFDM has to be extensible and guarantee the proper granularity, providing at the same time the enablers for data consistency, data safety, and proprietary data management.

Sacco et al (2011) conceived the VFDM as a set of XSD files (W3C, 2004c) defining the structure of the XML files that would be stored and managed by the VFM. This solution offers relevant advantages in terms of:

- Syntactic validation of the XML files according to the defined XSD files.
- Rich expressiveness since several default data types can be further extended and complex constraints and properties can be modelled.
- Possibility to integrate several XSD files within a single project.

However, the XSD technology alone is not suitable for knowledge representation and several flaws can be highlighted:

- No explicit characterization of data with their relations on a semantic level.
- Intra–document references are supported but inter–document references (cross-references) are poorly modelled, thus endangering referential consistency.
- Distributed data can be hardly managed.
- The integration of different knowledge domains can be cumbersome.

The presented considerations led to evaluate and finally adopt the Semantic Web technologies which offer key advantages to the whole VFF because they enables to:

- Represent a formal semantics.
- Efficiently model and manage distributed data.
- Ease the interoperability of different applications.
- Process data outside the particular environment in which it was created
- Exploit generic tools that can infer from and reason about an ontology, thus providing a generic support that is not customized on the specific domain.

The new semantic VFDM has been designed as an ontology (W3C, 2004a) by adopting the OWL language (W3C, 2004b). In particular, it defines all the *classes*, *properties* and *restrictions* that can be used to create the *individuals* to be stored in the data repository (Pillar II). Given the wide range and heterogeneity of the knowledge domains to be covered by the VFDM in the scope of VFF, it is necessary to integrate various knowledge domains as already highlighted by Colledani et al (2008) and Valente et al (2010) in previous related works. Therefore, the VFDM has been decomposed into macro areas (i.e. bricks), creating a hierarchical structure of sub-ontologies that have been named *Factory, Building, System, Resource, Process, Product, Strategy, Performance and Management*. This architecture allows decomposing the problem, downsizing its complexity while keeping a holistic approach. These sub-ontologies have been

developed by referring to the state-of-the-art technical standards available in the different domains, and in particular the Industry Foundation Classes (IFC2x3, 2006), STEP-NC (ISO 14649-10:2004), and ISA-95 (ISA-95).

4. VIRTUAL FACTORY MANAGER

This section presents the analysis of the requirements for the VFM (Sect. 4.1) and its proposed architecture (Sect. 4.2).

4.1 VFM REQUIREMENTS

The main goal of the VFM design and implementation consists in obtaining an open integration platform representing a common and shared communication layer between already existing and newly developed software tools to support the factory design and management.

The preliminary architecture of VFM proposed by Sacco et al (2011) was related to a VF Data Model based on the XSD/XML format. The adoption of an ontology-based representation of the VF Data Model in the VFF project has led to a re-design of the VFM where Semantic Web technologies have been exploited. In the new architecture previous basic requirements have been extended to include specific semantic functionalities:

- *Platform independent interfacing capabilities*. The VF modules are software tools developed by different vendors/organizations, with different programming languages, operating systems and HW architectures. The VFM has to interface all of them by providing its service in an open and "proper" way.
- *Management of concurrent access and data consistency*. Several software tools can access and/or modify partial areas of the factory data at different, and possibly overlapping, times. Therefore, the VFM is required to ensure that concurrent accesses occur without endangering the data integrity and slowing down the planning process to unacceptable levels.
- *Management of evolving Factory Data*. The VFM has to provide functionalities for managing the evolution and revision of the data related to complex entities like production systems, processes and products. A typical VFDM object is made by several files, depending on the sub-ontologies it refers to, as described in section 3. Hence, a coherent versioning mechanism must take into consideration the inter-document references between sub-ontologies.
- *Data safety* must be ensured in case of hardware failures or user errors.
- *Addition of customized functionalities*. Third party developers need an appropriate mechanism to enrich the set of functionalities provided by the VFM without impacting on its core.
- *Response time*. The interaction between VFM and the VF modules requires the support of communication mechanisms that are able to provide answers in an appropriate time frame.
- *A Semantic Web Endpoint* which enables stakeholders to query virtual factory models with the required level of granularity for a more efficient and selective data access.

Most of the above underlies the development of the previous version of VFM. For this reason the architecture of the new *semantic version* shares similar features with its predecessor. However the ability to support validation and queries of semantic data introduces novelty aspects in the overall design of VFM.

4.2 SEMANTIC VFM ARCHITECTURE

The architecture of the semantic VFM was designed to provide support to the required functionalities. Each solution implemented by the VFM is based on stable and well established technologies in order to obtain an overall system capable to respond to industrial needs of reliability. The resulting VFM architecture is shown in Figure 2 as an UML component diagram.

The functionalities of the VFM are exposed as web services that have been identified as a suitable and widely adopted solution to guarantee platform independent interfacing capabilities. The Application Server provides the front end for the exposure of VFM functionalities and takes care for the information transport of the VFM.

The Information Exchanging Platform (IEP) is the main component of the VFM and provides VF modules and plugins with a high level access to the two functional cores of VFM: the *Versioning Layer* and the *Semantic Layer*. It represents the preferred way (even if not the only one) to connect to the VFM, since it provides a complete set of methods for structured data retrieval and semantic validation, data locking mechanism and factory version management.

The Versioning Layer contains the *VF Data Repository* where all the shared data are stored. The evolution of the factory data is managed by the *Versioning System* that organizes and updates the set of virtual factory instances. The Versioning System guarantees the data safety as well, since it allows restoring an older version at any time, thus preventing data losses due to user errors. Moreover, rollback methods can be used in case of data inconsistencies due to broken connections or other

factors, always ensuring data safety. In particular, the locking mechanism exposed through the IEP helps to manage the concurrent access of the VF modules.

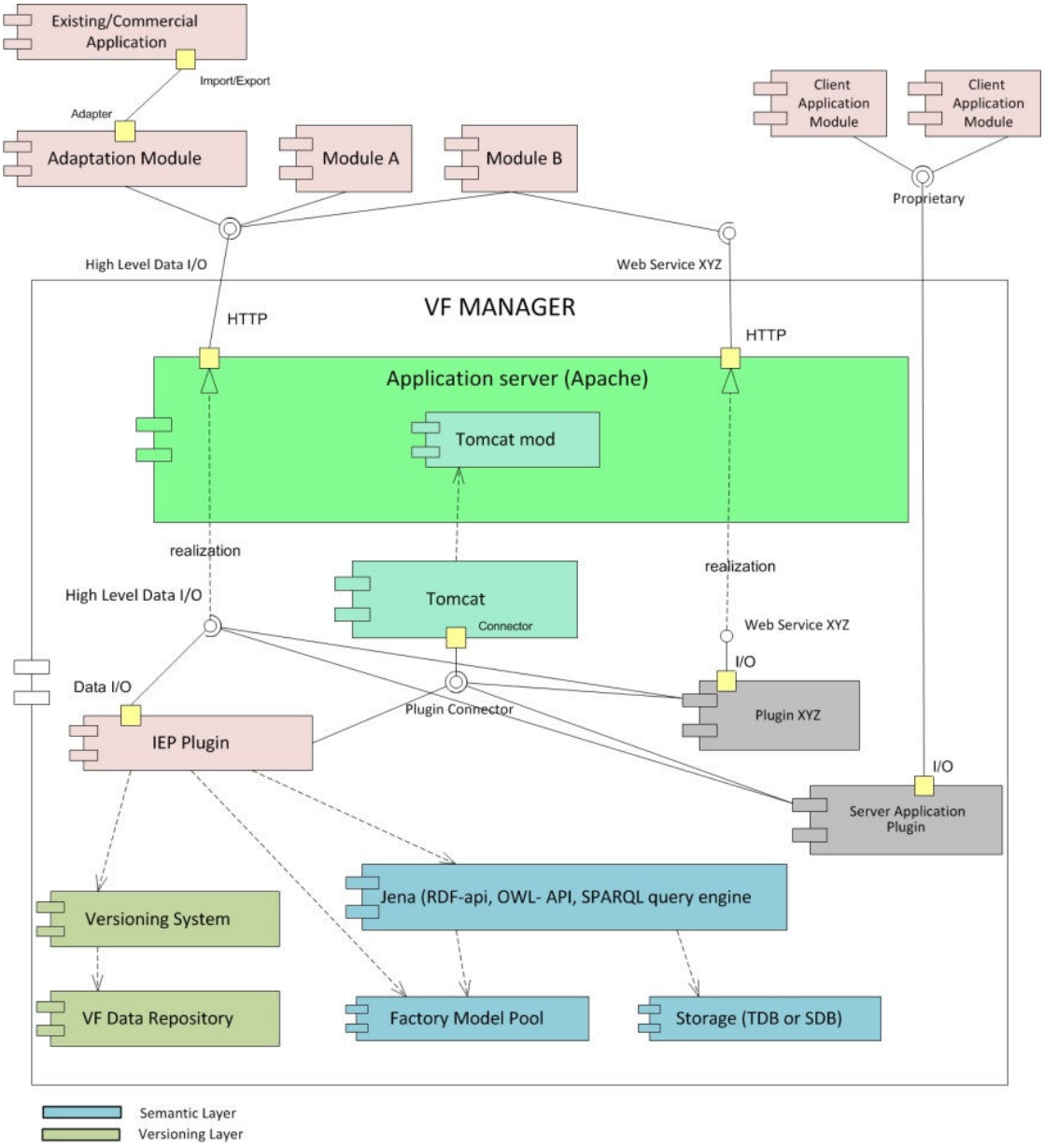

Figure 2 - Semantic VFM Architecture

The Semantic Layer is implemented by embedding in VFM one of the most common and reliable Semantic Web Frameworks: Jena (Jena, 2011). Through the IEP users can carry out semantic validations of VFF models using Jena functionalities directly on the server. Thanks to Jena, the IEP can also provide a *VFM SPARQL endpoint*. By starting a *Query Session* on data extracted from the VF Data Repository it is possible to perform SPARQL queries (W3C, 2008a). Through queries each module (or plugin) can select and aggregate information and be fed with exactly the data it needs for its business process. Model modifications can also be executed using the SPARQL Update language (W3C, 2008a). Modified models can then be serialised in output files in the same format used by the VF Data Model ontology (RDF/XML).

5. DECOUPLED VF MODULES

Currently many software applications, called VF Decoupled Modules, are under development and will interface the VFM for accessing the information kept in the data model.

Since the VFM is the centre of the data exchange among modules, it has been conceived with openness in mind to be able to deliver data to the large variety of tool involved in the factory planning

process. The decoupled modules range from completely new developments to integration of existing application, to off-the-shelf commercial software, characterized by different operating systems and development languages, among them Windows, Linux and Java, C++, Python.

5.1. REQUIREMENTS ORIGINATED BY THE VF MANAGER

Since the exposed functionality of the VFM is implemented as a web service (W3C, 2011), all the modules are required to implement a web service client according to the WSDL file (Booth and Liu, 2007) describing the published interface. Additionally, to address the issue that web services are intrinsically stateless, the VFM has implemented a few specific functions to keep track of the state of its clients. Therefore the decoupled modules need to actively support this mechanism. Finally, the data received from the VFM are in RDF/XML format (Beckett, 2004) and the utilization of third party libraries for the handling of that format is essential.

5.2. ARCHITECTURE

The listed common requirements lead to a similar overall module architecture which foresees a few predefined component.

The following diagram illustrates the generic architecture of a VF decoupled module with the mentioned components and a section of the VFM in the bottom part of the picture.

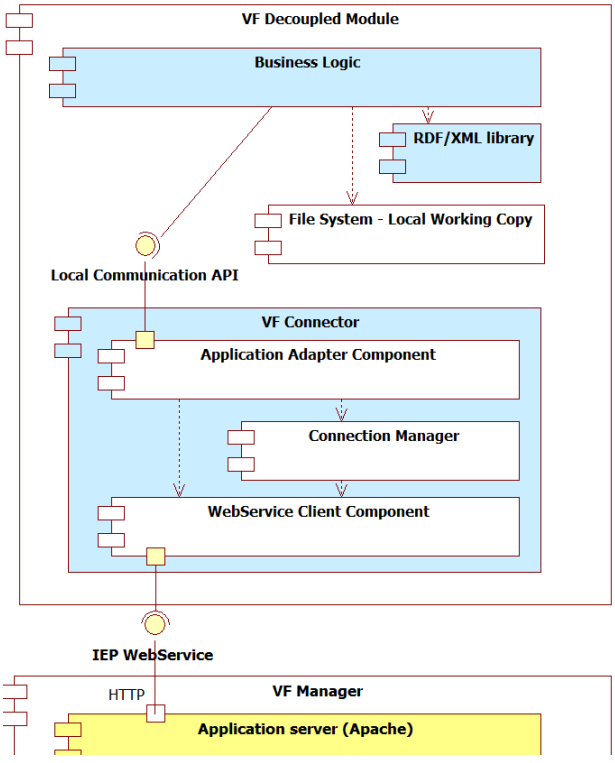

Figure 3 - Decoupled VF Module Architecture

5.2.1. VF Connector

Since all the decoupled modules will face common tasks related to the VFM connection, in order to avoid repeated development efforts among the VFF partners, a specific VF Connector for the most common development languages (C++, Java and Python) has been implemented.

The VF Connectors take care of the web service client implementation and the connection state mechanism.

5.2.2. RDF/XML Library

Each decoupled module will manage different parts of the data model in different ways. Nevertheless most of the results coming from the VFM are in form of RDF/XML streams so that the development effort will be reduced using already existing third party libraries conceived for RDF/XML data manipulation. The following table lists some of the most used open source libraries.

Table 1- RDF/XML Libraries

Library	Language	Link
Redland RDF Libraries	C with Python- Perl- PHP- Ruby- Interfaces	http://librdf.org/
RDFLib	Python	http://www.rdflib.net/
Jena RDF API	Java	http://jena.sourceforge.net/
Protegé API	Java	http://protege.stanford.edu/
Sesame OpenRDF	Java	http://www.openrdf.org

Semantic data handling obviously is more complex than the one required for XSD/XML-based models. Most of the VF Decoupled Modules are not *semantic applications* (Motta and Sabou, 2006). As such they access the VFDM semantic representation only to extract and modify “plain” data. Indeed this is one of the few disadvantages of the proposed semantic approach that can be mitigated only by fully exploiting the related technology to ensure the full integration of the four Pillars of VFF.

5.2.3. Business Logic

This part of the software is peculiar to each module and will be developed on top of the mentioned components.

Nevertheless it is possible to distinguish two substantially different functionalities; the *ad hoc* developed modules will provide a specific logic and expose it through a graphical user interface while the adaptation modules will provide the required interface for a seamless integration of existing commercial tools.

6. VF MANAGER PROTOTYPE

The Semantic VF Manager has been implemented on the basis of the previous prototype presented in Sacco et al (2011). Even if this version is a complete rewrite of the software, the main inspiring guidelines that have driven the first prototype release have not been changed. The choice of development based on an open source and cross platform architecture is still valid. This allows the deployment of the tools in real industrial scenarios where the VF Manager should be integrated into existing legacy intranet architectures. Having chosen to adopt technologies with proven reliability, cross platform compatibility and well known by IT personnel grants a smooth integration and successful operation inside most of the existing network configuration.

We hereby describe, for each of the main components of the architecture shown in Figure-2, the prototypal choices of the applied software tools. The Application Server was implemented with Apache HTTP Server, one of the most deployed and reliable HTTP servers (The Apache Software Foundation, 2011a). The Servlet Container was developed with Apache Tomcat (The Apache Software Foundation, 2011b), used in numerous large-scale, mission-critical web applications across a wide range of industries and organizations. Tomcat is paired with "Tomcat mod" to support integration with Apache HTTP server: this connector redirects the information received by the Apache Server to Tomcat, and therefore to the plug-ins. The Versioning Layer was developed by adopting Subversion (Collins-Sussman et al, 2004) that is an open source version control system widely used in the open source world. Access to the ontology model, i.e. creating, writing and reading models from OWL, has been implemented on top of the Jena framework (Jena, 2011) which is a proven library that implements Web Semantic in Java.

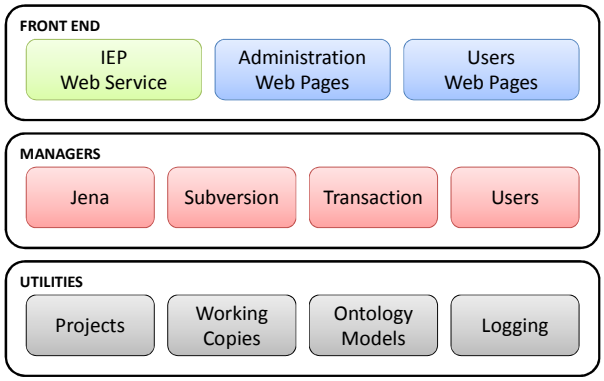

Figure 4 - Architecture of the VF Manager Prototype

The architecture of the VF Manager in Figure-4 highlights the internal division in tree main layers: front end, managers and utilities.

The *front end layer* consists of three components that expose the functionalities:

- *IEP* provides a SOAP Web Service to the VF modules.
- *Administration* lets the administrative personnel manage the user and the opened sessions using a web-based interface.
- *Users Pages* are web pages that a user can access to see his open session and to chat with other active users.

The *managers layer* groups most of the business logic of the VF Manager and is composed of the following components:

- *Jena* handles the ontology model using the Jena framework;
- *Subversion* is responsible for data storage and versioning using the SVNKit library (SVNKit, 2011);
- *Transaction* manages user sessions and commits, coordinating the Jena, Subversion and Users components;
- *Users* manages the user database, access control and sessions lists.

The *utilities layer* consists of components providing a common infrastructure to handle:

- Projects
- Working copies
- Ontology models
- Logging

All this components have been developed in Java and JSP (Java Server Pages) and deployed as Tomcat web application.

This prototype is an evolution of the software presented in Sacco et al (2011) and focuses on the implementation of a wider set of features:

- Data versioning
- SPARQL Query
- Locking
- Granularity
- Dependencies
- Web access

The previous prototype focused only on versioning and locking. A key feature introduced in this version is granularity, implemented with the concept of projects to represents the minimal independent unit (i.e. set of files) that can be locked. A project can have one or more dependency on other projects, allowing reusing of shared resources. A project can not only declare its dependencies – allowing a module to manually retrieve them - but can also automatically include them to build a

complete ontology model. This model can be queried using SPARQL language (W3C, 2008a). Finally in this prototype we have implemented the ability to access the VF Manager using an Internet browser: administrators and registered user can access some functionalities of the VF Manager without needing to use a module application.

Further development of the prototype is targeted at consolidating the robustness of the implemented features and at improving the functionalities that can be accessed from the web interface. This last development will eventually enable a user to access the data in the VF Manager without relying to any module, maybe allowing limited editing capabilities. Another important features that will be enabled by the web interface is the access from mobile devices, such as smart phones and tablets.

7. FLP – A DECOUPLED VF MODULE

In order to illustrate the interaction of decoupled modules with the VFM one among the several tools developed for the VFF project has been chosen, the Factory Layout Planner (FLP) (Ceruti et al, 2010). The FLP, together with other two applications (GIOVE Virtual Factory by ITIA-CNR and Visual Components with SimX adaptation module), was already involved in the feasibility demonstration of the former VFM (Sacco et al, 2011).

7.1. FUNCTIONALITY

FLP is a client/server application that enables the collaborative development of a factory layout.
The main functionality of the FLP consists in:

- 3D visual editing of the layout
- 3D visual editing of the building
- running Discrete Events Simulation (DES)

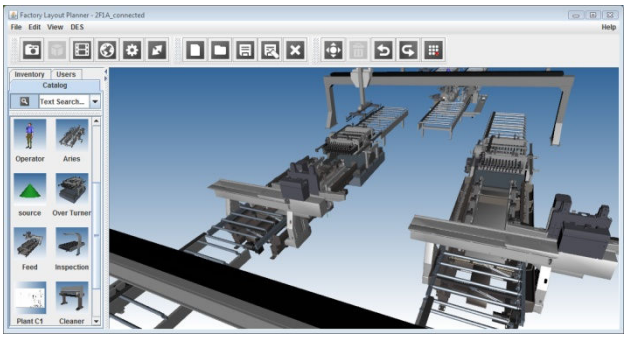

Figure 5 - FLP Main Window

The application is characterized by a two-level architecture with a fat client dealing with complex 3D models and real time requirements, and a server which acts as a synchronization manager and as VFM web client.

7.2. IMPLEMENTATION

FLP is an application written in Java. For handling the data received from the VFM in RDF/XML format, the third party library Jena (Jena, 2011) is used.

7.3. ACCESSED DATA

FLP will interact with different areas of the VFDM to exploit its functionality:

- the **multisite** information - the production plants of the enterprise [read-write]
- the **building** data [read-write]
- the **resources templates** - for every resource template (or type) a set of information (icon, VRLM file(s) for 3D representation, properties and further data) [read-only]
- the **layout** data (which contains for example the instantiated resources, their position and their property values) [read-write]
- the **production plans** and **processes data** to feed the DES engine [read-only]
- the **results of the DES** simulation [read-write]

7.4. SAMPLE USE CASES

The prototype of the VFM hosts a partial Data Model and enables the FLP to access data related to the layout of a factory, both resource types and instantiated resources. The FLP use cases treated here are related to those data.

7.4.1. Caching resources types (read only)

The FLP composes a layout by creating and placing into the 3D window instances of resources from the resources catalogue. The FLP itself does not modify the resource templates, which consists of resource type description data and typically very large 3D model files (those data are not subject to frequent changes). Given those assumptions, for performance purposes, the FLP maintains a local cache of the data by checking from time to time if a synchronization of the local cache is required.

The steps to accomplish this task consist in

- select the current revision of the sub-project "Resource Library"
- download the ontology file(s) containing the individuals
- load the file(s) into a Jena ontology model in order to access the content
- retrieve all the 3D files from the "resource catalogue" folder and store them locally

7.4.2. Layout planning (read-write)

The FLP user selects the current version of a layout with the purpose of modifying it. It must signal to the VFM its intention, so that other users do not

modify the same data and the Data Model remains consistent. Precondition for this use case is that the local catalogue of resources has been successfully synchronized for the selected project.

The additional steps (compared with use case 7.4.1.) required to accomplish this task consist in:

- start a transaction on the selected project
- display all the instances of resources defined in the project in a editable 3D view of the layout
- apply the modifications to the Jena ontology model and modify the local project file(s)
- send back the modified files to the VFM
- make the changes permanent and available to all others VFF users by committing the open transaction

8. CONCLUSIONS

The new approach driven by the Semantic VFDM and enabled by the Semantic VFM represents a step forward in the improvement of the Virtual Factory Framework and in particular towards the target of a fully integrated architecture of all the Pillars. Data individuals and their Semantics come now from the same coherent source and can be seen by different perspectives, according to the needs of the accessing clients (Knowledge Manager or Decoupled Modules).

A first prototype of the Semantic VFM exploring the potentials and the issues related to this new approach has been presented. In particular, the applied Semantic Web technologies represent the cornerstone to obtain a framework where the different stakeholders can effectively contribute in a harmonized way to the definition of the virtual factory along all the phases of its lifecycle.

In the coming months improved versions of the VFM will implement the defined architecture and fulfil the expected functionality. An increasing number of Decoupled Modules will interface the VFM and fill in the VF Data Repository with individuals according to the developed Virtual Factory Data Model, thus validating the new approach.

9. ACKNOWLEDGMENTS

The research reported in this paper has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement No: NMP2 2010-228595, Virtual Factory Framework (VFF).

REFERENCES

Beckett, D., “RDF/XML Syntax Specification (Revised)”, W3C, 2004, Retrieved: 15.06.2011, <http://www.w3.org/TR/rdf-syntax-grammar/>

Booth, D. and Liu, C.K., “Web Services Description Language (WSDL) Version 2.0 Part 0: Primer”, W3C, 2007, Retrieved: 15.06.2011, <http://www.w3.org/TR/wsdl20-primer/>

Bracht, U., Masurat, T., "The digital factory between vision and reality", *Computers in Industry*, Volume 56, Issue 4, 2005, pp. 325–333

Carroll, J.J., Dickinson, I., Dollin, C., Seaborne, A., Wilkinson, K. and Reynolds, D., “Jena: Implementing the Semantic Web Recommendations”, *Proceedings of the 13th international World Wide Web conference*, 2003, pp 74-83

Ceruti, I.F., Dal Maso, G., Ghielmini, G., Pedrazzoli, P. and Rovere, D., “Factory Layout Planner”, *ICE - 16th International Conference on Concurrent Enterprising*, 2010, Lugano, Switzerland

Chryssolouris, G., Mavrikios, D., Papakostas, N., Mourtzis, D., Michalos, G., Georgoulias, K., "Digital manufacturing: history, perspectives, and outlook", *Proceedings of the Institution of Mechanical Engineers Part B: Journal of Engineering Manufacture*, Volume 223, No. 5, 2008, pp.451-462

Colledani, M., Terkaj, W., Tolio, T. and Tomasella, M. “Development of a Conceptual Reference Framework to manage manufacturing knowledge related to Products, Processes and Production Systems”, In: Bernard A, Tichkiewitch S (eds), “Methods and Tools for Effective Knowledge Life-Cycle-Management”, Springer, 2008, pp 259-284.

Collins-Sussman, B., Fitzpatrick, B.W. and Pilato, C.M., “Version Control with Subversion”, 1st Edition, O'Reilly Media, Sebastopol, CA, 2004, p 320

Colombetti, M., “Ingegneria della conoscenza: modelli semantici”, 2010-11 Edition, Facoltà di ingegneria dell’informazione, Politecnico di Milano, Italy, 2011, p 42

Huang, G.Q., Simpson, T.W. Pine II, B.J., “The power of product platforms in mass customization”, *International Journal of Mass Customisation*, Vol. 1, No. 1, 2005, pp 1-13

ISA-95, “ISA-95: the international standard for the integration of enterprise and control systems”, ISA-95.com, Retrieved: 15.06.2011, <http://www.isa-95.com/>

Jena, “Jena – A Semantic Web Framework for Java”, SourceForge.com, 2011, Retrieved: 15.06.2011, <http://www.openjena.org/>

IFC2x3, “IFC2x3 Release”, buildingSmart, 2006, Retrieved: 15.06.2011, <http://buildingsmart-tech.org/specifications/ifc-releases/ifc2x3-release>

Mc Bride, B., “An Introduction to RDF and the Jena RDF API”, 2010, Retrieved: 15.06.2011, <http://jena.sourceforge.net/tutorial/RDF_API/>

Mc Carthy, P., “Search RDF data with SPARQL”, IBM, developerWorks, 2005, Retrieved: 15.06.2011, <http://www.ibm.com/developerworks/xml/library/j-sparql/>

Motta, E. and Sabou, M., “Next Generation Semantic Web Applications”, *1st Asian Semantic Web Conference (ASWC)*, 2006, Beijing, China

Sacco, M., Dal Maso, G., Milella, F., Pedrazzoli, P., Rovere, D. and Terkaj, W., “Virtual Factory Manager”, *HCI International*, 2011, Orlando, USA

Sacco, M., Pedrazzoli, and Terkaj, W., “VFF: Virtual Factory Framework”, *ICE - 16th International Conference on Concurrent Enterprising*, 2010, Lugano, Switzerland

SVNKit, “[Sub]Versioning for Java”, TMate Software, 2011, Retrieved: 15.06.2011, < http://svnkit.com/>

The Apache Software Foundation, “Apache HTTP Server Project”, The Apache Software Foundation, 2011, Retrieved: 15.06.2011, <http://httpd.apache.org/>

The Apache Software Foundation, “Apache Tomcat 6.0”, The Apache Software Foundation, 2011, Retrieved: 15.06.2011, <http://tomcat.apache.org/tomcat-6.0-doc/index.html>

Terkaj, W., Tolio, T., Valente, A., “Designing Manufacturing Flexibility in Dynamic Production Contexts”, In: Tolio, T. (ed) Design of Flexible Production Systems. Springer, 2009, pp 1-18

Tolio, T., Ceglarek, D., ElMaraghy, H.A., Fischer, A., Hu, S., Laperrière, L., Newman, S., Váncza, J., “SPECIES -- Co-evolution of Products, Processes and Production Systems”, *Cirp Annals-Manufacturing Technology 59 (2),* 2010, pp 672-693

Valente, A., Carpanzano, E., Nassehi, A. and Newman, S. T., “A STEP compliant knowledge based schema to support shop-floor adaptive automation in dynamic manufacturing environments”, *Cirp Annals-Manufacturing Technology 59 (1),* 2010, pp 441-444

W3C, “OWL Web Ontology Language - Use Cases and Requirements”, W3C, 2004a, Retrieved: 15.06.2011, <http://www.w3.org/TR/webont-req/#onto-def>

W3C, “OWL Web Ontology Language - Reference”, W3C, 2004b, Retrieved: 15.06.2011, <http://www.w3.org/TR/owl-ref/>

W3C, “SPARQL Query Language for RDF”, W3C, 2008a, Retrieved: 15.06.2011, <http://www.w3.org/TR/rdf-sparql-query/>

W3C, “SPARQL Update”, W3C, 2008b, Retrieved: 15.06.2011, <http://www.w3.org/Submission/SPARQL-Update/ >

W3C, “Web Services Activity”, W3C, 2011, Retrieved: 15.06.2011, <http://www.w3.org/2002/ws/>

W3C, “XML Schema Part 1: Structures Second Edition”, W3C, 2004c, Retrieved: 15.06.2011, <http://www.w3.org/TR/xmlschema-1/>

“ISO 14649-10:2004 Industrial automation systems and integration -- Physical device control -- Data model for computerized numerical controllers -- Part 10: General process data”