Virtual Factory Manager

Marco Sacco[1], Giovanni Dal Maso[2], Ferdinando Milella[3], Paolo Pedrazzoli[4], Diego Rovere[4], and Walter Terkaj[1]

[1] ITIA-CNR, Institute of Industrial Technologies and Automation, National Research Council, via Bassini, 15 - 20133 Milano, Italy
{marco.sacco,walter.terkaj}@itia.cnr.it
[2] Technology Transfer System S.r.l., via Pacini 15, 20131 Milano, Italy
dalmaso@ttsnetwork.com
[3] SimX Ltd., Furness House, Salford Quays, M50 3XA, Manchester, UK
f.milella@simx.co.uk
[4] ICIMSI, Institute of Computer Integrated Manufacturing for Sustainable Innovation, Galleria 2, Manno, Switzerland
{paolo.pedrazzoli,diego.rovere}@icimsi.ch

Abstract. The current challenges in manufacturing engineering are the integration of the product/process/factory worlds (data and tools) and the synchronization of their lifecycles. Major ICT players already offer all-comprehensive Product Lifecycle Management suites supporting most of the processes. However, they do not offer all the required functionalities and they lack of interoperability. An answer will be given by the development of a Virtual Factory Framework (VFF): 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) that acts as a server supporting the I/O communications within the framework for the software tools needing to access its data repository. The VFM will ensure data consistency and avoid data loss or corruption while different modules access/modify partial areas of the data repository at different times. Finally, an industrial case study will show the potentiality of the VFM.

Keywords: Virtual Factory, Interoperability, Reference Model.

1 Introduction

Manufacturing has to cope with a complex and evolving market environment. While the world crisis breaks the balance between demand and production, the global market pushes for a continuous change. Several critical aspects related to the 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 [1], while maintaining economies of scale and scope within the manufacturing processes [2]. 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 [3]. This synchronization can be empowered by digital technologies, as it is shown both by

R. Shumaker (Ed.): Virtual and Mixed Reality, Part II, HCII 2011, LNCS 6774, pp. 397–406, 2011.

industrial practice and academic scientific research. Indeed, the topic of Digital and Virtual Factory [4, 5, 6, 7] have been addressed by several research projects, such as METNET, SPAS, MANUVAR, EMI, EMERALS [8], VFTS [9], IRMA, DiFac [10] and Dorothy [11]. The Virtual Factory (VF) paradigm can assist a production environment by addressing various key issues like: (1) reduction of production times and material waste thanks to the analysis of virtual mock-ups, (2) development of a knowledge repository where people can find stored information in different versions, with both advisory role and support to the generation of new knowledge, (3) improvement of workers efficiency and safety through training and learning on virtual production systems, (4) creation of a collaboration network among people concurrently working on the same project in different places.

The complexity of the problem calls for support tools to effectively address all the phases of the factory lifecycle. Indeed, the ICT players (e.g. Siemens PLM, PTC and Dassault Systèmes) already offer all-comprehensive suites containing software tools that have been developed or acquired in the recent years. These tools deal with most of the factory planning, design and deployment phases. However, the current approaches still do not meet the demands of the industry and fail to provide all the required functionalities. One of the reason is the lack of interoperability. Moreover, Small and Medium Enterprises cannot afford the present expensive PLM software suites.

The previous analyses highlight the need of a novel framework for the VF enabling a step forward in the state of the art, by describing the factory as a whole consisting of processes, dependencies and interrelations, data and material flows [12]. This framework should guarantee the democratization of industrial systems for simulation and analysis, by reducing the total cost of ownership of a holistic virtual representation of the factory. In particular, the following aspects of democratization need to be properly addressed:

1. *Decrease of the investment and operating costs* that are currently associated with the commercial all-comprehensive software suites.
2. *Wider range of users*. The functionalities and usage of simulation and virtual factory technologies should be extended from the workstations/desktops of the engineers to the laptops of managers and marketing executives.
3. *Usage and management*. Democratization also means giving engineers and field technicians the ability to take advantage of simulation and virtualization of processes without relying on dedicated specialists.

This paper presents a framework for the VF that is carried out by the European project “Virtual Factory Framework” [13]. The framework is introduced in Section 2, whereas Section 3 focuses on the development of its main software components. Finally, Section 4 shows the results obtained with the first implemented prototypes.

2 Virtual Factory Framework

The Virtual Factory Framework (VFF) can be defined as “An integrated collaborative virtual environment aimed at facilitating the sharing of resources, manufacturing information and knowledge, while supporting the design and management of all the

factory entities, from a single product to networks of companies, along all the phases of the their lifecycles". The VFF architecture (see Fig. 1) is based on four Pillars: (I) *Reference Model*, (II) *Virtual Factory Manager*, (III) *Decoupled Functional Modules* and (IV) *Integration of Knowledge*. The key characteristics of the pillars are openness, scalability and easiness to plugin the decoupled software tools, thus reducing the *investment costs* compared to "all-in-one" software suites. Moreover, the VFF aims at promoting major *time and operating cost* savings, while increasing the performance in the design, management, evaluation and reconfiguration of new or existing factories.

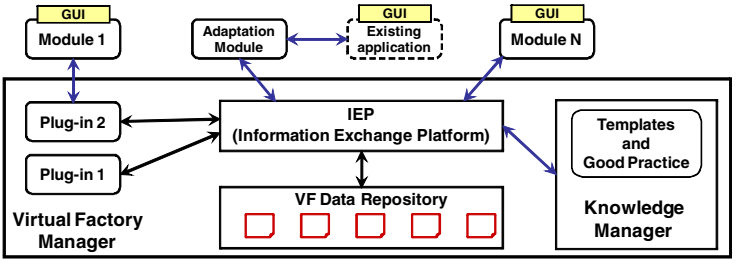


Fig. 1. Virtual Factory Framework architecture

The *Reference Model* establishes a coherent standard extensible Virtual Factory Data Model 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 Virtual Factory Manager (Pillar II) and used and updated by the Decoupled Functional Modules (Pillar III). Since the VFF has to deal with the interactions between several activities, it is necessary to take care of data evolution along the factory lifecycle phases as well.

The *Virtual Factory Manager* (VFM) is the core of the VFF and handles the common space of abstract objects representing the factory as defined by the common data model (Pillar I). The VFM orchestrates the decoupled functional modules and provides a controlled access to the different virtual factory instances. Considering the characteristics of the data model and the need to interface several decoupled modules, the structure of the VFM has been designed by adopting *star network* architecture as shown in Fig. 1. Next section will delve into the structure of the VFM by presenting its main characteristics and the development of the first prototype.

The *Decoupled Functional Modules* (named *VF modules*) are the software tools that implement the various methods and services to support the activities related to factory design, performance evaluation, management, production monitoring, etc. The VF modules can be located on a remote workstation or on the server where the VFM resides. Considering the scope of the VFF approach, the VF modules can be grouped into categories. For each category, different solutions can be adopted according to the specific needs and the availability of commercial applications. The integration of VF modules endowed with different functionalities and level of detail but insisting on the same factory representation will offer the possibility to reach a *wide range of users*.

Integration of Knowledge aims at supporting the modeling of complex systems and providing greater comprehension of the business processes. The Knowledge Manager (see Fig. 1) is responsible of the knowledge repository that is designed according to an ontology-based approach. A knowledge association engine will extract the knowledge from the repository by means of rule-based mechanisms and case-based reasoning techniques. The exploitation of embedded and constantly growing knowledge presented as good practice and templates (i.e. toolsets of pre-dealt problems) will enable the engineers to *easily use and manage* the virtual factory without specific expertise or experience.

3 Virtual Factory Manager

This section presents the analysis of the requirements for the VFM (Sect. 3.1) and its proposed architecture (Sect. 3.2). Finally, the VFM prototype is described (Sect. 3.3).

3.1 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 tool to support the factory design and management. This final goal leads to the definition of several requirements:

- *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 services 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.
- *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 the VFM and the VF modules requires the support of communication mechanisms that are able to provide answers in an appropriate time frame.

3.2 VFM Architecture

The architecture of the 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 Fig. 2 as an UML component diagram.

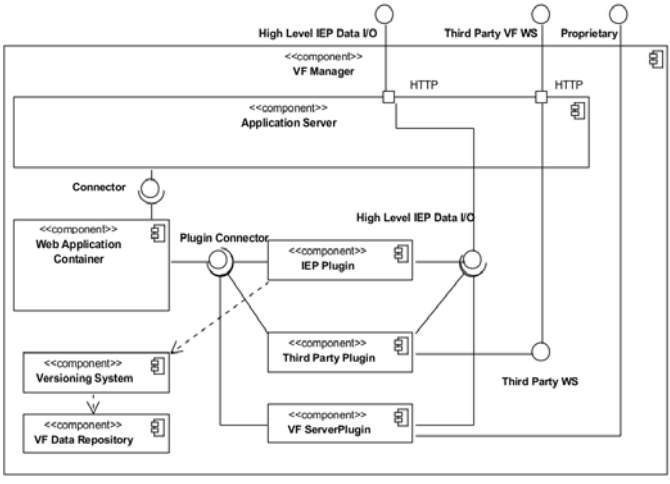


Fig. 2. UML component diagram of the VFM

The *VF Data Repository* is the central data repository where all the shared data will be 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 anytime, 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.

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 *Web Application Container* is the component of the architecture that provides a platform for implementing and publishing VFM functionalities.

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 data stored in the VFM. It represents the preferred (even if not the only one) way to connect to the VFM, since it provides a complete set of methods for structured data retrieval and validation, data locking mechanism and factory version management. In particular, the locking mechanism helps to manage the *concurrent access* of the VF modules.

A *Plugin* is a component that enables the VFM to be scalable by hosting server side third party software, thus *adding customized functionalities*. Server side software packages provide their own specific services interfacing the containing framework

that ensures local optimized access to the *IEP* and support to a container already configured for exposing methods as web services.

The interaction between the VFM and the VF modules mainly consists in an exchange of data streams only after specific requests (e.g. download/upload of data, locking of data, etc.) are made by the VF module. This kind of interaction requires no "real-time" *response* from the VFM. Nevertheless, if some module has specific requirements on the type of interaction (e.g. for response time or for data encryption), the implementation of a server side plugin with a dedicated proprietary interface is always possible and supported by the VFM structure.

3.3 VFM Prototype

The implementation process of the VFM prototype was driven by the adoption of open source platforms while following the architecture in Fig. 2. This decision was taken to obtain a completely open solution that can be developed and maintained by different actors. Java was chosen as software platform because of the availability of high-level and reliable tools that enable the application in real industrial scenarios.

The *VF Data Repository* was developed as a file system where all the data are stored on the server in form of files. The adoption of a file-based system instead of a Database Management System (DBMS) is justified by its flexibility and the possibility to apply a versioning system. Most of the data are stored in XML files [14] that can be validated by a set of XSD (XML Schema Definition) [15] files representing the the VF Data Model. Besides XML files, the file system can host files of any kind (e.g. binary files for graphical and geometric representations).

The *Versioning System* was developed by adopting Subversion [16] that is a widespread open source version control system. Its original purpose is to maintain versions of software source code and text documents, but it can be used to track changes in any kind of file and it is suitable for the VFM needs since it is efficient when applied to text-based files as the XML files are.

The *Application Server* was implemented as an Apache HTTP Server [17] that is an open source modular web server and one of the most deployed HTTP servers. Moreover, Apache is well known for being reliable and for supporting most of the programming languages and platforms.

The *Web Application Container* was developed using Apache Tomcat [18] that is an open source project of the Apache Software Foundation. It powers several large-scale and mission-critical web applications for a wide range of companies and organizations. Tomcat can be paired with "Tomcat mod" that is a supporting component required to forward the information received by the Apache Server to Tomcat, and then to the plugins.

The *IEP* component was implemented as a Tomcat web application and its functionalities are exposed as a set of *cross-platform* web services based on SOAP (Simple Object Access Protocol) [19], thus enabling any VF module to use them. The IEP prototype provides both automated versioning and locking mechanisms to prevent data inconsistency thanks to "check-out" and "commit" operations.

4 Testing the VFM Prototype

This section explores the potentiality of the VFF and in particular of the VFM by showing how different software tools can interoperate while addressing the same industrial problem. In particular, the test case is focused on the design of a factory layout for a Romanian company (Compa S.A.) playing in the automotive market. A reduced version of the final VF Data Model was developed as an XSD file and three software tools were deployed in the VFF as VF modules: *GIOVE Virtual Factory* [20], *Factory Layout Planner* [21], and *3DCreate* by *Visual Components Oy* [22].

GIOVE Virtual Factory (*GIOVE VF*) is a virtual reality collaborative environment aimed at supporting the factory layout design. Machines, operators and other resources can be selected from a library and placed in the 3D scene of the virtual factory. The virtual environment can schematically display performance measures that are provided by simulators and/or monitoring tools. Thanks to its user-friendly interface, GIOVE VF enables the collaboration between managers, experts and also workers in an intuitive way.

Factory Layout Planner (*FLP*) is a client/server application that enables the collaborative development of a factory layout thanks to three key features: the 3D visual editing of the layout, the possibility to act on the same layout in a distributed environment, the ability to perform Discrete Events Simulation (DES) on the layout. The collaboration on the layout can be both remote and local; the former allows user distributed all over the world to cooperate in the layout creation, the latter allows users to act on the same device at the same time on a common model. This functionality can be achieved thanks to the integration of multi-touch tables.

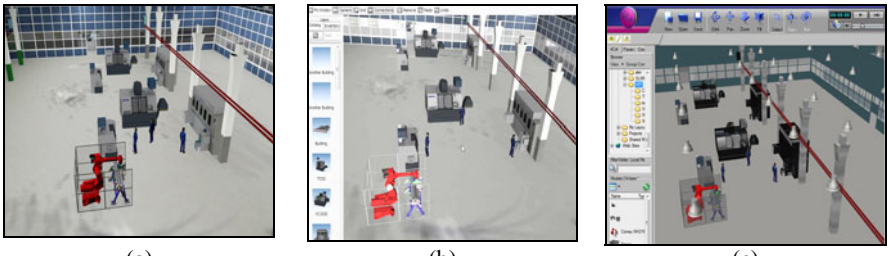
3DCreate by *Visual Components Oy* performs material flow and robotic simulation on the same platform for simulation and visualization of complete manufacturing systems. 3D equipment models can be created with increasing level of details to a point where they represent real factory counterparts in look and behaviors. New factory layouts are created from a catalog by simply snapping equipment models together. The user can run simulations and perform a large number of general layout validations, like collision detection, resource utilizations, cycle times, etc.

The interoperation between the previously described software tools represents an interesting test bed for validating the VFM concept because:

- The tools were developed with different programming language and can operate on different OS and platforms, thus enabling the validation of the VFM "universality": i.e. GIOVE VF was developed in C++, whereas FLP in Java. Visual Components, developed in C++, was connected to the VFM through Python.
- The tools share some functionalities (e.g. visualization of a 3D factory layout), thus enabling a validation of the VFM functionalities.
- The tools have complementary functionalities (e.g. navigation in a 3D environment, DES and kinematic simulation provided by GIOVE VF, FLP and Visual Components, respectively), thus showing the benefits of interoperability.

During the test each software tool had a different user and was run on a different computer. The three software tools were connected to a server hosting the VFM and exposing its web services. The basic VFM functionalities were successfully tested since all the three VF modules could access the web services exposed by the VFM,

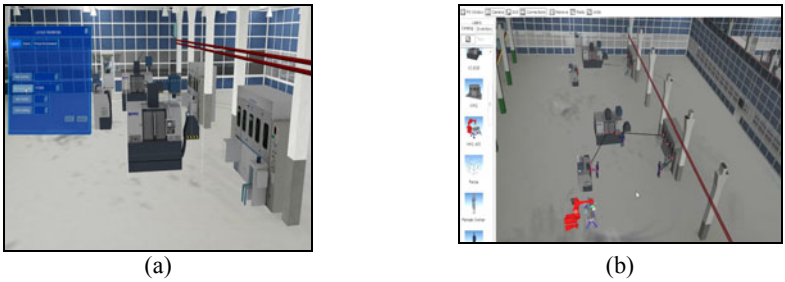
thus meeting the requirement of *platform independent interfacing capabilities*. The XML files in the VF Data Repository were correctly imported and interpreted by the three tools thanks to the common data model. Figure 3 shows how the three tools offer only a slightly different graphical representation of the same factory view.



(a) (b) (c)

Fig. 3. Same factory view offered by (a) GIOVE VF, (b) FLP and (c) Visual Components

The interoperation between the VF modules was tested by incrementally modifying a factory layout in a collaborative way, starting from a simple layout and then gradually increasing its complexity by adding production resources and using functionalities provided by the VF modules. Every step was performed through a check-out/modify/commit cycle. The VF modules have to access the factory data in a selective way, leaving untouched data that the VF module does not use and/or is not capable of using. For example, after the GIOVE VF user created a draft factory layout (Fig. 4.a), the FLP user modified the layout by adding new machines and operators and creating connections between the resources (Fig. 4.b), so that a DES could be run afterwards. The modifications implemented by the FLP user were finally checked and correctly visualized by the GIOVE VF user, thus demonstrating how the VFM is able to meet the requirement of guaranteeing the *data consistency* across VF modules.

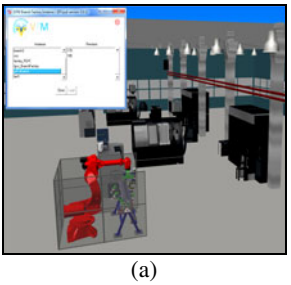


(a) (b)

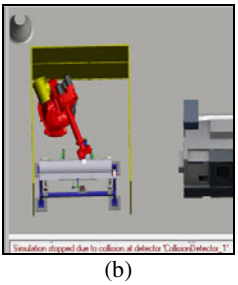
Fig. 4. (a) The GIOVE VF user and (b) the FLP user modify the same factory layout

Already existing factory instances can be used as a starting point to develop new layouts or when a stakeholder wants to evaluate "what if" scenarios without overwriting the existing data and upsetting the lifeline of a factory instance. A new instance can be created from an existing one at any stage of its lifeline thanks to the "branch-out" functionality provided by the IEP. For example, the Visual Components

user created a new factory instance starting from the version previously saved by the GIOVE VF user, thus showing how the VFM can *manage evolving factory data*. Then the Visual Components user modified the layout by adding a fence around the robot (Fig. 5.a) and checked potential collisions of the robot by means of a kinematics simulation (Fig. 5.b). Finally, the *management of concurrent access* was tested too. If a VF module tries to access a factory version that is currently locked by another user, then the request is denied and the factory data can be downloaded in read-only mode.



(a)



(b)

Fig. 5. A Visual Components user (a) adds a fence to the layout and then (b) performs a kinematics simulation of the robot to detect possible collisions with the fence

5 Conclusions

The proposed framework and the VFM provide a concrete answer to the requirements of interoperability to address the product/process/factory lifecycle. The VFM prototype and the feasibility studies presented in this paper represent a first cornerstone for future developments.

The use of XSD/XML has guaranteed to have a pre-defined syntactic structure for the data (XML) that can be validated according the XSD definitions. Moreover, XSD/XML represents an expressive technology where several default data-types can be extended and complex constraints and properties can be modeled. However, the complex processes associated with the virtual factory require also the support of knowledge and the characterization of data with their relations on a semantic level. This is hardly given by the current data representation based on XSD, since inter-document references (cross-references) are poorly modeled, thus endangering the referential consistency and the management of distributed data. Therefore, the possibility to adopt an ontology-based representation of data and exploit the Semantic Web technologies for the VFM will be evaluated.

In the coming months more complex industrial cases will be addressed and more software tools will be integrated to present a complete solution for the whole factory lifecycle. Finally, further functionalities will be added to the VFM.

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). The authors would like to thank COMPA S.A. (Sibiu, Romania) for kindly providing the industrial case information.

References

1. Huang, G.Q., Simpson, T.W., Pine II, B.J.: The power of product platforms in mass customization. International Journal of Mass Customisation 1(1), 1–13 (2005)
2. Terkaj, W., Tolio, T., Valente, A.: Designing Manufacturing Flexibility in Dynamic Production Contexts. In: Tolio, T. (ed.) Design of Flexible Production Systems, pp. 1–18. Springer, Heidelberg (2009)
3. 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), 672–693 (2010)
4. Zöllner, M., Keil, J., Behr, J., Gillich, J., Gläser, S., Schöls, E.: Coperion 3D – A Virtual Factory on the Tabletop. In: Proceeding of 5th INTUITION International Conference: Virtual Reality in Industry and Society (2008)
5. Ding, J., Wang, Y., Chen, K.: An Interactive Layout and Simulation system of Virtual Factory. Applied Mechanics and Materials 20-23, 421–426 (2010)
6. Yang, S., Ahn, B., Seo, K.: Development of a prototype customer-oriented virtual factory system. The International Journal of Advance Manufacturing Technology 28(9-10), 1031–1037 (2006)
7. Zhai, W., Fan, X., Yan, J., Zhu, P.: An Integrated Simulation Method to Support Virtual Factory Engineering. International Journal of CAD/CAM 2(1), 39–44 (2002)
8. Jain, S., Choong, N.F., Aye, K.M., Luo, M.: Virtual factory: an integrated approach to manufacturing systems modeling. International Journal of Operations & Production Management 21(5/6), 594–608 (2001)
9. Kazlauskas, E.J., Boyd, E.F., Dessouky, M.M.: The Virtual Factory Teaching System (VFTS): Project Review and Results. In: Proceedings of ED-MEDIA 2002 World Conference on Educational Multimedia, Hypermedia & Telecommunications (2002)
10. Sacco, M., Redaelli, C., Cândea, C., Georgescu, A.V.: DiFac: an integrated scenario for the Digital Factory. In: Proceedings of 15th International Conference on Concurrent Enterprising (2009)
11. Pedrazzoli, P.: Design Of customeRdRivenshOes and multisite factory – DOROTHY. In: Proceedings of 15th International Conference on Concurrent Enterprising (2009)
12. Pedrazzoli, P., Sacco, M., Jönsson, A., Boër, C.: Virtual Factory Framework: Key Enabler For Future Manufacturing. In: Cunha, P.F., Maropoulos, P.G. (eds.) Digital Enterprise Technology, pp. 83–90. Springer, Heidelberg (2007)
13. Sacco, M., Pedrazzoli, P., Terkaj, W.: VFF: Virtual Factory Framework. In: Proceedings of 16th International Conference on Concurrent Enterprising, Lugano, Switzerland (2010)
14. Extensible Markup Language (XML), `http://www.w3.org/XML/`
15. XML Schema Part 1: Structures 2nd edn, `http://www.w3.org/TR/xmlschema-1/`
16. Collins-Sussman, B., Fitzpatrick, B., Pilato, M.: Version Control with Subversion (2008), `http://svnbook.red-bean.com/index.en.html`
17. Apache, HTTP Server Project, `http://httpd.apache.org/`
18. Apache Tomcat 6.0, `http://tomcat.apache.org/tomcat-6.0-doc/index.html`
19. SOAP Specifications, `http://www.w3.org/TR/soap/`
20. Viganò, G.P., Greci, L., Sacco, M.: GIOVE Virtual Factory: the digital factory for human oriented production systems. In: Proceedings of the 3rd International CARV Conference, Munich, Germany, pp. 748–757 (2009)
21. Ceruti, I.F., Dal Maso, G., Ghielmini, G., Pedrazzoli, P., Rovere, D.: Factory Layout Planner. In: Proceedings of 16th International Conference on Concurrent Enterprising, Lugano, Switzerland (2010)
22. 3DCREATE Visual Components, `http://www.visualcomponents.com/Products/3DCreate`