Virtual Factory Data Model to support Performance Evaluation of Production Systems

Walter TERKAJ[a], Marcello URGO[b]
[a] Istituto Tecnologie Industriali e Automazione (ITIA), Consiglio Nazionale delle Ricerche (CNR)
[b] Dipartimento di Ingegneria Meccanica, Politecnico di Milano

Abstract. The performance evaluation of manufacturing systems is a critical and difficult task to be addressed throughout the factory life-cycle phases, including the early design, detailed design, ramp-up, reconfiguration, and monitoring. An efficient and effective performance evaluation may have a relevant impact on the profitability of an industrial company. This paper addresses the application of a data model for virtual factories to the performance evaluation problem, aiming at exploiting the interoperability with other software tools to continuously update the virtual representation of a manufacturing system, so that accurate estimations can be obtained. A test case is described and then used to check the viability of the proposed approach in the case of Discrete Event Simulation (DES) based on a commercial software tool like Arena.

Keywords: Performance Evaluation, Discrete Event Simulation, Data Model, Ontology, Manufacturing Systems, Virtual Factory

1. Introduction

The design of manufacturing systems is a complex task strictly related to manufacturing strategy decisions having an impact on a long time horizon (usually more than two years) and involving a major commitment of financial resources [**1**]. For instance, strategic decisions may regard the number of plants or facilities to be built, their size and their location, the variety of products to be manufactured, the manufacturing technology to be used and, within a plant, the number and type of production resources, the characteristics of the transportation and handling systems, the degree of automation. The complexity of these decisions and their importance from the point of view of the profitability of capital investments emphasizes the need to have formal and structured approaches to evaluate the performance of a manufacturing system.

Usual performance indicators in a manufacturing context can be the production volumes, the quality of the output, the incurred cost, etc. In addition, more detailed performance indicators may be calculated, e.g. the utilization of production resources, the average flow time of products, the average level of the work in progress. Different models can be used to address specific types of analysis and levels of detail while modeling a manufacturing system to evaluate its performance. In the field of discrete part manufacturing, two main approaches are in common use:

- *Analytical models* using mathematical or symbolic relationships to provide a formal description of the system [**2**] [**3**]. The model is then used to derive an explicit expression of a performance measure or, in most of the cases, to define an algorithm or a computation procedure able to calculate the performance indicators.
- *Simulation models* represent the events occurring in a manufacturing system in its operation by a sequence of steps that are executed in a computer program [**4**] [**5**]. This sequence of steps is generated with respect to a set of rules modeling the behavior of the system. Therefore the characteristics and relationships between the elements in a manufacturing systems can be described in detail. However, the higher is the detail level and the higher is required computational effort. If a simulation model is run for a sufficiently long time, then proper statistics can be collected and performance indicators can be estimated.

Simulation models enable the representation of an higher level of details, thus providing more accurate estimates of the manufacturing system behavior compared to analytical models. However, to reach this level of details, also a more detailed formalization of the manufacturing system is needed. Simulation modeling of manufacturing systems usually relies on commercial software tools (e.g. Arena, Simio, Plant Simulation, Visual Components, etc.) providing an integrated environment to describe the system and its behavior in terms of relationships and rules and, in addition, to deal with the generation of random values and the collection of the statistics.

Performance evaluation tools can be more effective if they are based on a virtual representation of the manufacturing system that is continuously updated during both the design and operational/execution phase, thus guaranteeing an overall coherence of the obtained results. Moreover, the generation of a simulation models and/or analytical model can be time-consuming and it would be beneficial if this activity could be as much automated as possible. The resulting need of interoperability between performance evaluation tools and tools supporting the design and management of real industrial systems can be met by an extended framework enabling:

- the cooperation among different actors with different competences and expertise in the design and management of a factory based on common definitions and a shared virtual representation of its components linking different manufacturing domains while guaranteeing their coherence;
- the management and update of a huge amount of manufacturing data made available through standard and interoperable interfaces.

The development of a framework for the interoperability between software tools supporting factory processes is currently carried out by the European project “Virtual Factory Framework” [**6**]. 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” [**7**]. The VFF architecture is based on three main pillars:

- Virtual Factory Data Model (VFDM), i.e. a coherent, standard, extensible, and common data model for the representation of factory objects related to production systems, resources, processes and products [**8**].

- Virtual Factory Manager (VFM), i.e. the manager of a shared data repository containing factory data that can be accessed and modified by all the software tools integrated in the framework [**9**] [**10**].
- decoupled Virtual Factory modules, i.e. the software tools that are able to communicate with the VFM to retrieve and send shared data formalized according to the VFDM (e.g. [**11**]).

This paper focuses on development and enhancement of the VFDM for modeling a generic manufacturing system and then evaluating its performance. Section 2 gives an overview of the current state of the art on data models for manufacturing systems and presents the VFDM solution. Section 3 describes a test case representing a production line. Section 4 delves into the problem of evaluating the performance of a manufacturing system formalized according to the VFDM; in particular Discrete Event Simulation by means of the commercial software tool Arena is addressed. Finally, conclusions are drawn in Section 5.

2. Modeling Manufacturing Systems

2.1 State of the Art

Several scientific contributions and proposed technical standards have faced the problem of developing a holistic and complete data model for representing manufacturing systems, both considering tangible (e.g. machine tools, part types to be produced, etc.) and intangible (e.g. process plans, production logics, etc.) aspects.

Among the available technical standard, ANSI/ISA-95 [**12**] is an international standard for developing an automated interface between enterprise and control systems. This standard has been developed for applications in all industries and in all sorts of processes, like batch processes, continuous and repetitive processes. ISA-95 aims at providing both consistent terminology and information models as well consistent operations models. B2MML (Business To Manufacturing Markup Language) [**13**] is an XML implementation of the ANSI/ISA-95 and consists of a set of XML schemas [**14**] that implement the data models in the ISA-95 standard.

According to ANSI/ISA-95 standard, a manufacturing process can be modeled using the *ProcessSegment* class. The *ProcessSegment* class can represent a single step in a manufacturing process or a whole process through composition. The *ProcessSegment* class is linked to further classes to characterize the process, e.g. the needed equipment (*EquipmentSegmentSpecification* class), the personnel (*PersonnelSegmentSpecification* class) and the material (*MaterialSegmentSpecification* class). Furthermore, precedence relations between different process steps can be defined thanks to the *ProcessSegmentDependency* class. The *EquipmentSegmentSpecification* class allows the user to specify the pieces of equipment needed for the execution of a process step and how the equipment. ANSI/ISA-95 standard enables the user to freely define customized properties that can be attached to most of the classes representing processes and production resources. However, such flexibility can be a major drawback from the interoperability point of view. Indeed, if two users adopt ANSI/ISA-95, they still have to agree on the definition of the object properties before being able to exchange data characterized by a proper semantic. Furthermore, ANSI/ISA-95 does not provide a complete support for

modeling physical data such as the placement and shape representation of objects in the manufacturing system (e.g. a machine tool).

A different approach in the modeling of manufacturing process is offered by the Process Specification Language (PSL) standard [**15**]. PSL is an ontology providing a way to formally describe a process and its characteristics. The ontology has been developed at the National Institute of Standards and Technology (NIST) and has been approved as an international standard in the document (ISO 18629). The PSL ontology grounds on a set of axioms of first order logic written in CLIF (Common Logic Interchange Format) and organized in a core set together with extensions. The core provides the definition of an activity and its occurrence related to a time variable. The extensions enable the modeling of the execution through states, the definition of logical expression constraining the execution of the activities, and the capability of modeling resource and their usage by the execution of the activities. Grounding on an ontology, the PSL standard provides a robust and reliable framework to formalize the knowledge related to a process and guarantee an adequate level of interoperability. However, this standard is still scarcely adopted in the industrial domain, probably because of the perceived complexity at the enterprise level.

The Industry Foundation classes (IFC) standard by buildingSMART [**16**], partially based on STEP standard [**17**], represents an open specification for *Building Information Modeling* (BIM) data that is exchanged and shared among the various participants in a building construction or facility management project. The IFC standard is available as an EXPRESS schema specification [**18**] and is structured as a set of schemas that are grouped into four layers: Resource layer (i.e. general purpose or low level concepts/objects), Core Layer (where the most abstract concepts of the model are defined), Interoperability Layer defining concepts or objects common to two or more domains, and the Domains/Application Layer. The standard was mainly conceived for Architectural Engineering Construction (AEC) industry domains (e.g. Building Controls, Structural elements, Structural Analysis, etc.) and therefore provides most of the definitions needed to represent tangible elements of a manufacturing systems. Furthermore, generic definitions of intangible characteristics (e.g. processes, work plans, etc.) are provided, so that its data structures can be specialized for other industrial domains, such as the manufacturing domain.

2.2 Virtual Factory Data Model

The Virtual Factory Data Model (VFDM) of the VFF project is based on already existing technical standards and extends their definitions to represent the characteristics of a manufacturing system in terms of the products to be manufactured, the manufacturing process they must undergo and the resources entitled to operate the different manufacturing operations [**8**]. The VFDM is mainly based on the IFC standard release IFC2x4 RC2 [**19**] that was translated into a set of ontologies by adopting the Semantic Web approach [**20**]. Indeed, the XSD/XML technology [**14**] was considered at first, but it is not suitable for knowledge representation, explicit characterization of data with their relations on a semantic level, and management of distributed data, thus endangering referential consistency. On the other hand, the Semantic Web approach offers the possibility to represent formal semantics, merge ontologies dealing with different domains, efficiently model and manage distributed data, and ease the interoperability between different applications.

The *Entities* in the IFC standard are mapped to *OWL Classes* in the VFDM. Most of the classes derived from IFC are specializations of two fundamental classes named *IfcTypeObject* and *IfcObject*, both being subclasses of *IfcObjectDefinition*. The former class is the generalization of any thing or process seen as a *type*, the latter seen as an *occurrence*. *OWL individuals* of class *IfcObject* may be linked with a corresponding individual of class *IfcTypeObject.*

IfcTypeObject has the following subclasses: *IfcTypeProduct*, *IfcTypeProcess*, *IfcTypeResource*. *IfcTypeProduct* represents a generic object type that can be related to a geometric or spatial context (e.g. manufactured products, machine tools, transport systems, etc.). *IfcTypeProcess* defines a generic process type that can be used to transform an input into output (e.g. assembly operation, machining operation, etc.). *IfcTypeResource* represents the information related to resource types needed to execute a process. A resource represents the "use of thing".

IfcObject has the three main subclasses (i.e. *IfcProduct*, *IfcProcess*, *IfcResource*) that represent an occurrence of the corresponding type modeled by the subclasses of *IfcTypeObject.*

The previously described generic classes can be exploited to model a wide range of manufacturing systems while taking into consideration both physical and logical aspects. The subclasses of *IfcTypeObject* can be used to specify the designed characteristics of a manufacturing system, e.g. the part types to be produced (as individuals of *IfcTypeProduct*), the process plans (as individuals of *IfcTypeProcess*), the required type of production resources (as individuals of *IfcTypeResource*). On the other hand, the subclasses of *IfcObject* can be used to represent the execution phase of a manufacturing system by defining the workpieces in process (as individuals of *IfcProduct*), the actually executed operations (as individuals of *IfcProcess*), and the usage of production resources (as individuals of *IfcResource*).

The relations between the processes and resources can be formalized as shown in Figure 1 where the boxes represent classes and the arcs represent property restrictions linking classes according to the Manchester OWL Syntax [**21**]. Moreover, Figure 1 shows how system design data (upper part of the figure) can be linked with system execution data (lower part of the figure).

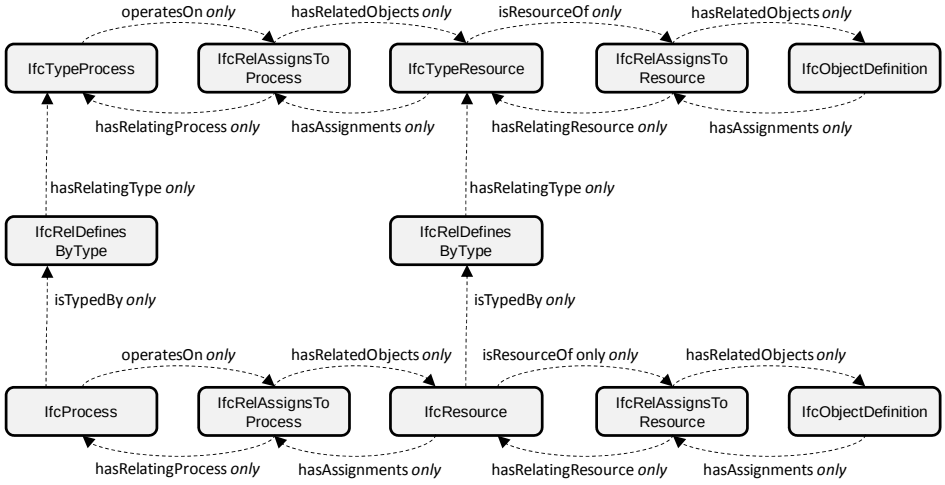

Figure 1. Relations between process and resource classes in the VFDM.

During the manufacturing system design/planning phase, the resource types needed by a process type can be specified by means of the objectified relationship class *IfcRelAssignsToProcess*, whereas the resource providers (as individuals of class *IfcObjectDefinition*) can be linked to a resource type thanks to the class *IfcRelAssignsToResource*.

During the manufacturing system execution phase (both real and simulated), occurrences of processes and resources can be created while referring to specific types defined during the design phase thanks to the class *IfcRelDefinesByType*.

As described by Terkaj et al. [**8**], the VFDM specializes some classes of the IFC standard for the manufacturing domain, paying attention in particular to the *type* classes *IfcTypeProduct*, *IfcTypeProcess*, *IfcTypeResource* and the corresponding *occurrence* classes *IfcProduct*, *IfcProcess*, *IfcResource*.

VffProcessType and *VffProcess* are defined as subclass of *IfcTypeProcess* and *IfcProcess*, respectively, to model generic transformation processes that, provided a given input, obtains a certain output according to certain rules and using a specified set of resources, i.e. a recipe. A process can be described as a whole or can be decomposed into subprocesses thanks to the class *IfcRelNests*. *VffProcessType* and *VffProcess* are further specialized to represent manufacturing, assembly, maintenance and handling processes. Moreover, precedence constraints between the processes can be defined by means of the objectified relationship class *IfcRelSequence*, whereas input and output entities of a process can be linked by using the classes *IfcRelAssignsToProcess* and *IfcRelAssignsToProduct*, respectively.

VffProductionResourceType and *VffProductionResource* are subclasses of *IfcTypeResource* and *IfcResource*, respectively, modeling a generic resource used in a factory (and its production systems). These classes are further specialized to represent equipment resources, material resources, and human resources, respectively.

In the VFDM the classes *VffMachineryElementType* and *VffMachineryElementType* have been defined as subclasses of *IfcTypeProduct* and *IfcProduct*, respectively, to represent generic pieces of machinery equipment.

Finally, specific property classes (e.g. *VffProcessProperties*, *VffMachineryElementProperties*) have been created to properly characterize processes, resources and machinery elements.

3. Test Case on Production Line

This section presents a test case representing a production line to show how the VFDM can be employed to create factory projects and use them with different digital tools. The test case consists of four ontologies that instantiate the VFDM classes, thus exploiting the data distribution empowered by the Semantic Web approach: three factory libraries (i.e. *VffLibrary01*, *VffLibrary02*, *VffLibrary17*) and one main factory project (i.e. *VfProductionLine04*). All these ontologies import the set of VFDM ontologies.

VffLibrary01 ontology defines a production site and a building.

VffLibrary02 ontology defines five machine types (as individuals of class *VffMachineryElementType* (i.e. *MtA*, *MtB*, *MtC*, *MtD*, *MtE*). Each machine type is associated with two possible shape representations in VRML and 3DS format.

VffLibrary17 ontology defines a part type as individual of class *VffWorkpieceType* (i.e. a subclass of *IfcTypeProduct*) and a possible process plan to obtain a final product

from a raw piece. The process plan named *processPlan01* is defined as an individual of class *VffManufacturingProcessType* (i.e. a subclass of *VffProcessType*) and decomposed into five process segments (as individuals of class *VffManufacturingProcessType*) characterized by a processing time and a predefined sequence. Moreover, each process segment requires a specific type of production resource and the processing time is modeled as an exponential distribution (see Table 1).

Table 1. Process planning.

Individual of VffManufacturing ProcessType	**Description**	**Required resource type as individual of VffProductionResourceType**	**Stochastic Processing time distribution**
processPlan01	Process plan	N/A	Exponential(0.033)
DR01	Drilling operation	drillingRes01	Exponential(0.033)
ML01	Milling operation	millingRes01	Exponential(0.02)
ML02	Milling operation	millingRes02	Exponential(0.02)
QC01	Quality control	qualityControlRes01	Exponential(0.033)
GR01	Grinding operation	grindingRes01	Exponential(0.025)

VfProductionLine04 ontology contains the factory project that imports and enriches the data provided by the three libraries. The factory project defines the units of measurement, the representation context and world coordinate system where the production site and the building imported from *VffLibrary01* are placed. One production line is designed and placed in the building of the factory. The production line consists of seven machines (as individuals of *VffMachineryElement*) that are typed by the machine types defined in *VffLibrary02* (see Table 2) and characterized by a shape representation and a placement. The production line is designed to process the part type defined in *VffLibrary17* and is thus organized into five production stages. Each needed production resource type can be provided by one or more machinery element as shown in Table 2. An example of relations between the individuals defined in the test case is shown in Figure 2 where the boxes represent individuals (identified by their local URI and class) and the arcs represent object properties linking the individuals. In particular, it is shown that the process segment *ML02* requires the resource type *MillingRes02* that can be provided by the machine type *MtC* (i.e. *MS02* or *MS03*) or by the specific machine *MS04*.

Table 2. Machinery elements.

individual of VffMachineryElement	**Related individual of VffMachineryElementType**	**Description**	**Provided resource type as individual of VffProductionResourceType**
DS01	MtA	Drilling machine	drillingRes01
MS01	MtB	Milling machine	millingRes01
MS02	MtC	Milling machine	millingRes02
MS03	MtC	Milling machine	millingRes02
MS04	MtB	Milling machine	millingRes02
CS01	MtD	Quality control machine	qualityControlRes01
GS01	MtE	Grinding machine	grindingRes01

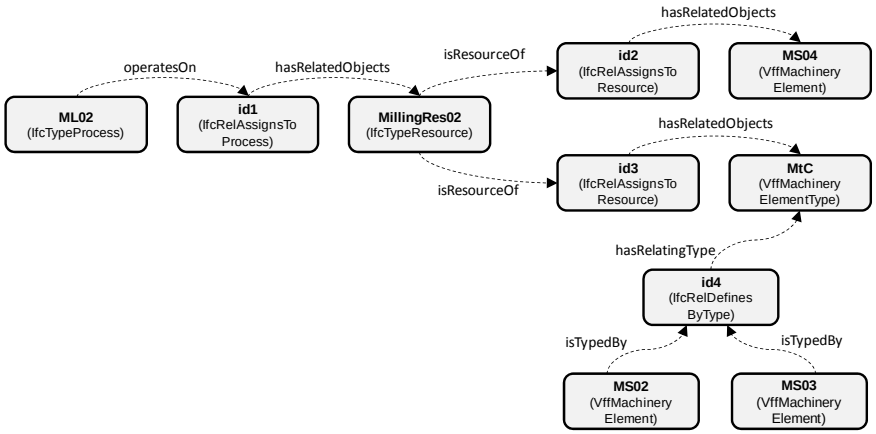

Figure 2. Relations between process type, resource type and machinery element.

Figure 3. 3D visualization of the manufacturing system represented in the test case.

The presented test case has been serialized in RDF/XML files [**22**] and can be uploaded/downloaded to/from a shared data repository that is made available by a VFM installation, so that the contained factory project and libraries can be accessed by any VF module that is integrated in the VFF framework thus. For instance, Figure 3 shows a visual representation of the factory project made by a VF module named GIOVE Virtual Factory [**11**].

4. Discrete Event Simulation

The applicability of the VFDM to model a manufacturing system and its behavior, aiming at evaluating its performance, has been validated focusing the attention on the case of Discrete Event Simulation (DES) and taking as a reference the test case presented in the previous section.

The capability of generating simulation models in an automatic (or semi-automatic) way has been often considered one of the great challenges in the simulation of manufacturing systems [**23**] [**24**]. In such kind of approaches, a simulation model is generated from a data source using algorithms for creating the model and proper interfaces to interact with a specific simulation environment. The automatic generation of a simulation model answers to the need of speeding up the overall time required to build a simulation model and, in addition, should also reduce the time needed to verify a model by decreasing the time required to debug the code.

Within the area of the simulation of manufacturing systems, similar issues have been also addressed in the literature, e.g. by Lorenz and Schulze [**25**], Randell and Bolmsj [**26**], and Mueller et al. [**27**]. Most of the presented approached are characterized by one or more of the following drawbacks:

- the work is strictly focused on a specific manufacturing sector (e.g. semiconductors [**27**]);
- lack of universal validity;
- limited level of automatism that can be actually reached.

The design of the VFDM was driven by the need of providing a way to unambiguously describe a generic manufacturing system regardless the specific application and, hence, to address some of the lacks of the approaches already proposed in the literature.

4.1 Discrete Event Simulation using Arena

Among the great number of available general-purpose commercial off-the-shelf (COTS) simulation packages, Arena by Rockwell Automation [**28**] is one of the most used both in the academic and industrial world for applications in the manufacturing field [**29**] [**30**].

An Arena model is built by dragging modules into the model window and connecting them to define the flow of entities through the model. An example is shown in Figure 4 where parts are generated in the *Generate Parts* block on the left and then flow to the *Machine Part* block representing the execution of a certain set of operations.

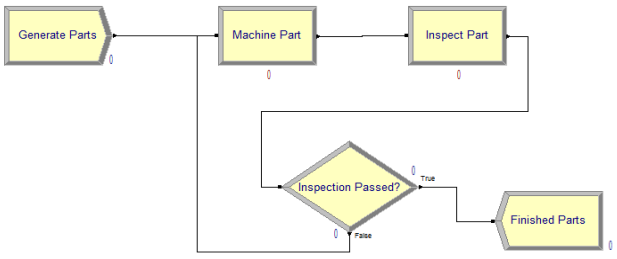

Figure 4. A simple simulation model in Rockwell Arena.

As shown in Figure 5, a set of resources can be invoked to operate the defined operations. In this case, each part entering the block asks for the resource *Machine 1* that has been already defined in the Arena model. A different and more flexible way of defining a process or a sequence takes advantage of the capability of defining sequences. A sequence consists of an ordered list of stations that an entity will visit.

For each station in the sequence, values may be assigned to attributes and variables. Moreover, using sequences it is possible to assign different routing to different type of entities, i.e. part types. Each station in the sequence is referred to as a step (or jobstep) and can be characterized by specific attributes (e.g. the processing time).

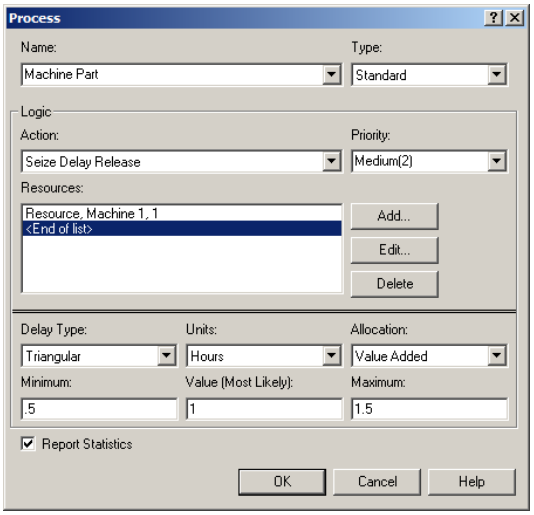

Figure 5. The process definition window in Rockwell Arena.

4.2 Arena and VFF

A DES simulator based on Arena can exploit the interoperability enablers offered by VFF only if it becomes a VF module (see Sect.1), thus being able to access and understand the contents of the shared data repository where factory projects and libraries are formalized according to the VFDM. The information stored into a VFDM-compliant project can be used to automatically generate an Arena simulation model only if the Arena data structures are properly mapped to the VFDM classes. This mapping has been implemented by a software component named *Arena-VF Connector* that is attached to Arena and works as a client exploiting the services offered by the VFM. The *Arena-VF Connector* has been developed in C++ language and can import/export ontologies serialized in RDF/XML format. The *Arena-VF Connector* makes use of the *VF Connector C++ Library* that is based on the Redland C libraries [**31**] and provides functionalities to parse, create and modify the ontologies thanks to an internal map between OWL classes/restrictions and C++ classes/methods. The instances of C++ classes are used as handlers of the ontology individuals to support and ease the binding between the factory project individuals and the internal data structure of Arena. The *Arena-VF Connector* makes use of the COM interface provided by Arena to automatically generate Arena models.

The development of a *Arena-VF Connector* requires a deep analysis because Arena represents a manufacturing system according to proprietary data structures and the relationships between the processes and resources are formalized in a different way compared to the VFDM. In particular, Arena requires each step of a process to be explicitly assigned to a station, thus preventing (grounding on the traditional modeling) the opportunity of defining a group of entities that can be used as resources and postponing the actual assignment of the parts to different stations at run time. The

definition of a sequence in Arena allows the modeling of the process steps, but each process steps must be directly linked to a station or a group of station of the same type. To cope with this limitation, the assignment of parts to stations at run time must be explicitly managed in Arena.

Taking as a reference the example in Figure 2, the manufacturing system model can be translated into an Arena model as shown in Figure 6. The resource type (e.g. *millingRes02*) required by a specific process step (e.g. the milling operation *ML02*) in the sequence is mapped to a *Station* block and the assignment to the available objects providing the needed resource (e.g. the machines *MS02, MS03, MS04)* is explicitly managed through a tree of *Branch* blocks with as many leaves as the number of machine occurrences that can execute the process step. On the leaves of the branch tree the parts to be manufactured are routed to the specific machine using a *Route* block after recording the destination in an attribute of the *Assign* block. Instead of *Branch* blocks, different methods can be used to model specific assignments policies as well. Finally, the machine occurrences (i.e. *MS02, MS03, MS04)* are mapped to *Station* blocks (see the bottom of Figure 6) that are followed by a traditional sequence of Arena blocks, i.e. a *Seize* block allotting the machine, a *Delay* block initialized with the processing time and a *Release* block freeing the machine. Then the manufactured part can be routed to the following process step in the sequence (*SEQ* keyword in the *Route* block).

Arena also offers the opportunity of modeling identical machines (e.g. *MS02* and *MS03*) as a single resource with cardinality greater than one. However, by adopting this option the dispatching of parts to the identical machines is internally managed by Arena according to predefined policies that cannot be directly controlled.

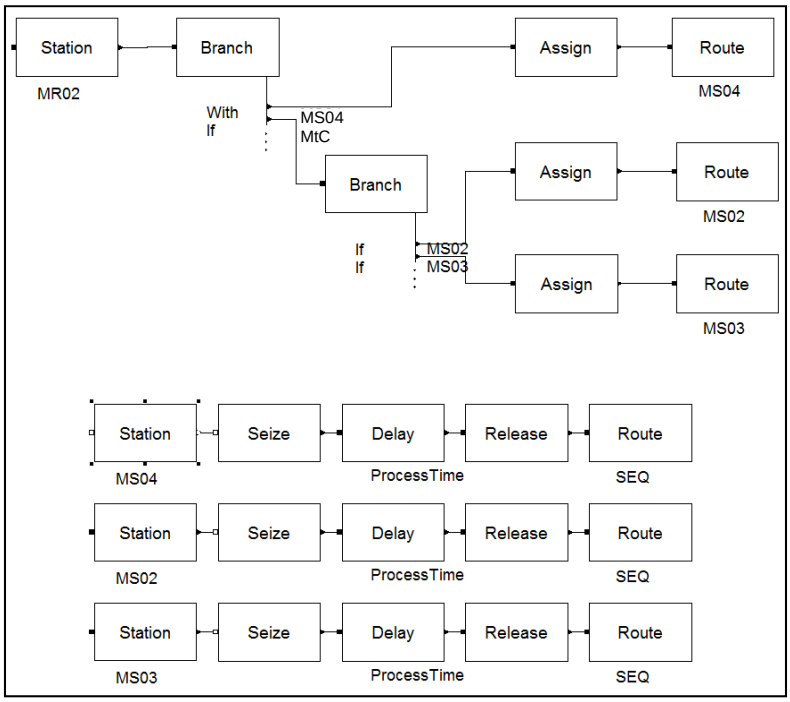

Figure 6. Automatically generated Arena model

Further blocks can be added to an Arena simulation model for collecting specific statistics regarding the involved resources. These statistics can be formalized according to the VFDM by using the definitions imported from the IFC standard. For instance, the *IfcResourceTime* class offers a set of attributes to store the time-related information associated to a resource, e.g. the start and finish time for the assigned workload and the percentage usage during the considered time horizon. The time-related attributes can specify scheduled or actual values, thus showing how the VFDM can be used to support factory planning, performance evaluation and factory monitoring activities.

The performance of the considered manufacturing system has been evaluated in terms of utilization of the different resources and flow time of the parts. The results have been also validated against a simulation model built manually and representing the same manufacturing system (Figure 7).

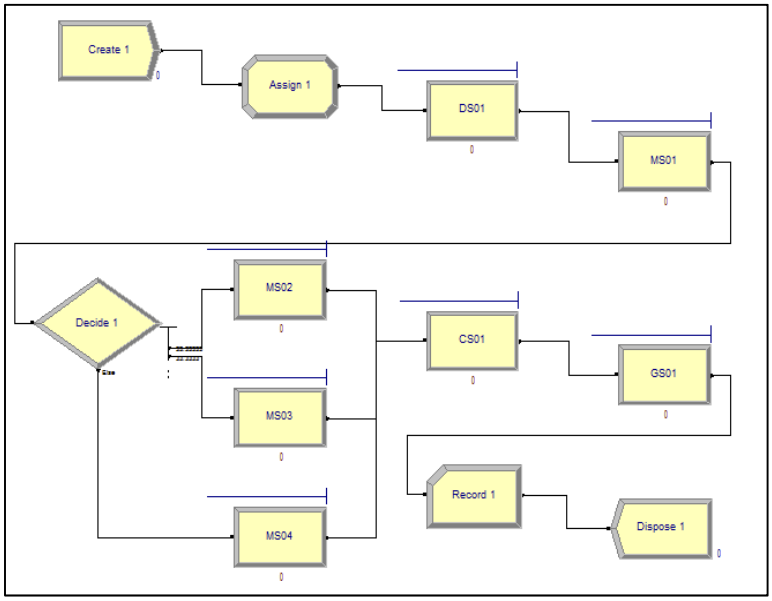

Figure 7. Manually generated Arena model

Table 3: Simulation results.

Performance indicator (average)	**Automatically generated model**	**Manually generated model**	**Confidence interval (99%) on the mean of the difference between the two results**
Utilization DS01	60.15 %	60.13 %	[-0.41,0.45]
Utilization MS01	60.20 %	59.94 %	[-0.16, 0.69]
Utilization MS02	33.49 %	33.05 %	[-0.94, 0.87]
Utilization MS03	33.30 %	33.53 %	[-0.31, 0.81]
Utilization MS04	33.44 %	33.52 %	[-1.03, 0.86]
Utilization CS01	59.99 %	59.82 %	[-0.79, 1.13]
Utilization GS01	80.26 %	79.63 %	[-0.12, 1.38]
Flow time	0.107 [hours]	0.104 [hours]	[0.00, 0.01]

Table 3 reports the results for 10 simulation runs of length 10 days with a warm up of one day, for both the automatically and the manually generated simulation models. The last column in Table 3 reports the 99% confidence intervals for the mean of the difference between the results of the two simulation models. All the confidence intervals contain the value 0, hence, the difference can be considered equal to 0 and, consequently, the two simulation models provides the same results demonstrating the validation of the automatically generated simulation model.

5. Conclusions

This paper has presented a data model for representing virtual factories, in particular aiming at modeling the complex relationships between physical and logical entities of a manufacturing system. It was shown how the adoption of a shared data model can enhance the interoperability between software tools supporting the design, management and performance evaluation of the factories.

Further developments of the data model are needed to better represent the production logics characterizing a manufacturing system so that the generation of a simulation model can be automated as much as possible. Moreover, the accuracy of the generated simulation models will be improved if the common data model is used to formalize the data coming from the shop-floor, thus closing the loop between the real factory and its virtual representation.

In this paper the VFDM has been used mainly to support interoperability, however further research can be carried out to exploit the enablers of the Semantic Web approach to perform reasoning and enrich the knowledge about specific manufacturing contexts.

Finally, the applicability of the VFF approach needs to be further tested by integrating more software tools for performance evaluation into the framework. Such integration will be supported by the development of programming libraries helping the implementation of customized versions of *VF Connector*.

Acknowledgements

The research reported in this paper has been funded by the European Union Seventh Framework Programme (FP7/2007-2013) under the grant agreement No: NMP2 2010-228595, Virtual Factory Framework (VFF) and the grant agreement No: 262044, VISION Advanced Infrastructure for Research (VISIONAIR). The authors would like to thank COMPA S.A. (Sibiu, Romania) for kindly providing information for representing the test case.

References

[1] W. Terkaj, T. Tolio, and A. Valente, "Designing Manufacturing Flexibility in Dynamic Production Contexts," in Design of Flexible Production Systems.: Springer, ch. 1, pp. 1-18.

[2] M. Colledani and T. Tolio, "A Decomposition Method to Support the Configuration/Reconfiguration of Production Systems," CIRP Annals - Manufacturing Technology, vol. 54, no. 1, pp. 441-444, 2005.

[3] M. Colledani, F. Gandola, A. Matta, and T. Tolio, "Performance evaluation of linear and non-linear multi-product multi-stage lines with unreliable machines and finite homogeneous buffers," IIE Transactions, vol. 40, no. 6, pp. 612-626, 2008.
[4] M. Bruccoleri et al., "Testing," in Design of Flexible Production Systems, T Tolio, Ed.: Springer, 2009, pp. 239-293.
[5] G. Pedrielli, M. Sacco, W. Terkaj, and T. Tolio, "Simulation of complex manufacturing systems via HLA-based infrastructure," Journal Of Simulation, To be published 2012.
[6] VFF, Holistic, extensible, scalable and standard Virtual Factory Framework (FP7-NMP-2008-3.4-1, 228595). [Online]. http://www.vff-project.eu/
[7] M. Sacco, P. Pedrazzoli, and W. Terkaj, "VFF: Virtual Factory Framework," in Proceedings of ICE - 16th International Conference on Concurrent Enterprising, Lugano, Svizzera.
[8] W. Terkaj, G. Pedrielli, and M. Sacco, "Virtual Factory Data Model," in Proceedings of 2nd OSEMA (Ontology and Semantic Web for Manufacturing) Workshop, 2012.
[9] M. Sacco et al., "Virtual Factory Manager," in Virtual and Mixed Reality - Systems and Applications. Proceedings of International Conference, Virtual and Mixed Reality 2011 held as Part of HCI International 2011, Orlando, FL, 2011, pp. 397-406.
[10] G. Ghielmini et al., "Virtual Factory Manager of Semantic Data," in Proceedings of DET2011 7th International Conference on Digital Enterprise Technology, Athens, Greece, 2011.
[11] G.P. Viganò, L. Greci, S. Mottura, and M. Sacco, "GIOVE Virtual Factory: A New Viewer for a More Immersive Role of the User During Factory Design," in Digital Factory for Human-oriented Production Systems, L., Redaelli, C., Flores, M. Canetta, Ed.: Springer, 2011, pp. 201-216.
[12] International Society of Automation. ISA-95: the international standard for the integration of enterprise and control systems. [Online]. http://www.isa-95.com/
[13] The Organization for Production Technology. (2011) Business To Manufacturing Markup Language (B2MML). [Online]. www.wbf.org/associations/12553/files/B2MML-BatchML%20v0500%20Schemas-Word-PDF.zip
[14] W3C. (2004, October) XML Schema Part 1: Structures Second Edition. [Online]. http://www.w3.org/TR/xmlschema-1/
[15] National Institute of Standards and Technology. (2008) Process Specification Language (PSL). [Online]. http://www.mel.nist.gov/psl/
[16] buildingSMART. IFC Overview. [Online]. http://buildingsmart-tech.org/specifications/ifc-overview
[17] International Organization for Standardization, ISO 10303 - Industrial automation systems and integration -- Product data representation and exchange.
[18] International Organization for Standardization, ISO 10303-11:2004 Industrial automation systems and integration -- Product data representation and exchange -- Part 11: Description methods: The EXPRESS language reference manual, 2004.
[19] buildingSMART. Industry Foundation Classes - IFC2x Edition 4 Release Candidate 2. [Online]. http://buildingsmart-tech.org/ifc/IFC2x4/rc2/html/index.htm
[20] W3C. (2009) OWL 2 Web Ontology Language - Document Overview. [Online]. http://www.w3.org/TR/owl2-overview/
[21] W3C. (2009) OWL 2 Web Ontology Language Manchester Syntax. [Online]. http://www.w3.org/TR/owl2-manchester-syntax/
[22] W3C. (2004) RDF/XML Syntax Specification (Revised). [Online]. http://www.w3.org/TR/REC-rdf-syntax/
[23] S. Bergmann and S. Strassburger, "Challenges for the Automatic Generation of Simulation Models for Production Systems," in Proceedings of the 2010 Summer Simulation Multiconference, Ottawa, Canada, 2010, pp. 545-549.
[24] J. W. Fowler and O. Rose, "Grand Challenges in Modeling and Simulation of Complex Manufacturing Systems," SIMULATION: The Society for Modeling and Simulation International, vol. 80, no. 9, pp. 469–476, 2004.
[25] P. Lorenz and T. Schulze, "Layout based model generation," in Proceedings of the 27th conference on Winter simulation (WSC '95), 1995, pp. 728-735.
[26] L. G. Randell and G. S. Bolmsjo, "Database driven factory simulation: a proof-of-concept demonstrator," in Proceedings of the 33nd conference on Winter simulation, 2001, pp. 977-983.
[27] R. Mueller, C. Alexopoulos, and L.F. McGinnis, "Automatic generation of simulation models for semiconductor manufacturing," in Proceedings of the 39th conference on Winter simulation, 2007, pp. 648-657.
[28] Rockwell Automation , Arena User's Guide, Version 12.00.00, 2007.
[29] Law and A. M., Simulation modeling and analysis, 4th ed.: McGraw-Hill, 2007.
[30] W.D. Kelton, Simulation with Arena, 5th ed.: McGraw-Hill, 2006.
[31] D. Beckett. Redland RDF Libraries. [Online]. http://librdf.org/