A Core Reference Model for Applicable Reconfigurable Manufacturing Systems

Pascal André, Olivier Cardin

A Core Reference Model for Applicable Reconfigurable Manufacturing Systems

Pascal André and Olivier Cardin

Abstract Reconfigurable Manufacturing Systems (RMS) have emerged as a promising approach to address the dynamic demands and uncertainties in modern manufacturing. However, the lack of a comprehensive and universally accepted definition of the notion of configuration in RMS poses challenges for researchers and practitioners. This research article aims to bridge this gap by suggesting a core reference model for RMS, allowing to exhibit a generic modelling of configurations. The article critically examines existing definitions and approaches to configuration, providing a comprehensive overview of the different dimensions and components of configuration in RMS. By addressing the lack of a consistent understanding of configuration in RMS, this article contributes to the development of a shared knowledge base and paves the way for further advancements in the field. It also highlights the importance of establishing a clear and holistic definition of configuration for effective implementation and utilization of RMS in practice.

Key words: Reconfigurable Manufacturing Systems, Metamodel, Holon, Concerns, Design pattern

1 Introduction

In the face of rapidly changing market demands, increasing product complexity, and technological advancements, manufacturers are seeking innovative solutions to enhance their agility, adaptability, and competitiveness. Reconfigurable Manufacturing Systems (RMS) have emerged as a promising approach to meet these challenges by providing a flexible and responsive manufacturing environment. However, despite the growing interest and adoption of RMS, there remains a critical gap in the understanding and definition of the notion of configuration within these systems. Con-

Pascal André · Olivier Cardin
LS2N UMR CNRS 6004 - University of Nantes and IUT de Nantes – 2, rue de la Houssinière F-44322 Nantes Cedex, France, e-mail: name.surname@ls2n.fr

figuration, as a fundamental concept in RMS, encompasses the arrangement, organization, and adaptation of system components, processes, and resources to meet specific manufacturing requirements. It encompasses the ability to modify and reorganize the system's physical, logical, and operational attributes to accommodate changing product specifications, production volumes, and market conditions. The configuration of an RMS influences its adaptability, scalability, and overall performance, thereby playing a crucial role in achieving operational excellence and competitive advantage.

Currently, there is a lack of a comprehensive and universally accepted definition of configuration in the context of RMS. This gap hampers researchers and practitioners in effectively understanding, designing, and implementing RMS. The objective of this article is to bridge the existing gap by suggesting a core reference model for RMS. The ultimate objective is to establish a foundation for a unified and comprehensive definition of configuration in RMS. To do so, this article will examine the different aspects and components of configuration, targeting modular design, adaptability, flexibility, and reconfigurability.

The findings of this research aims at contributing to the development of a shared knowledge base for researchers and practitioners in the field of RMS. A clear and holistic understanding of the notion of configuration will enable more effective design, implementation, and utilization of RMS in practice. It will also facilitate the development of standardized methodologies, tools, and frameworks for configuring RMS to meet evolving manufacturing needs.

The primarily targeted methodologies are related to a research project (RODIC project, funded by the French National Research Agency - ANR) where the focus is specifically on the evaluation phase of a given potential configuration. The configuration can therefore be roughly defined as a set of modules, arranged in a given manner with given parameters. In this phase of evaluation, several actions have to be executed:

- let the user define its configuration in a computable form
- verify the validity of the configuration
- define the test scenario that will be executed
- evaluate the performance indicators of the configuration in this scenario
- present the indicators in an understandable form to the user.

This list exhibits the need to have different perspectives on the notion of configuration all along the evaluation phase. Therefore, it seems valuable to connect all those perspectives with a single reference model, able to capture all the characteristics of a configuration in a RMS context.

The paper is organised as follows. We recall the background information in Section 2. Section 3 expresses the main principles of applicable reference architectures which are put in practice in the reference model we propose in Section 4. The architecture is modular to be customized in various contexts of RMS and then illustrated in a simple case in Section 5. In conclusion, we draw perspectives for a larger integration and development of the tool in an actual manufacturing context.

2 Background

3 Requirements

This sections discusses requirements for RMS reasoning and applicability. Most belong to the *Stream #2–analysis of RMS features* of Bortolini et al.'s survey [5]: modularity, integrability, diagnosibility, convertibility, customisation and scalability.

① **Conceptual models for RMS** The survey of Kayser et al. [11] pointed out that detailed reference architecture are under-specified to be applicable, there miss detailed models to design the software applications of the manufacturing systems, leaving such decisions to software delivers. As mentioned in [5], conceptual models for RMS are a missing but promising research stream. In this article, we will the widespread UML standard notation. To improve the modelling quality, we refer to the *core model principles* of [2]: meaningful notation, type/instance distinction, recursive aggregation, aggregation protocols, schematic specialisations.

② **Modularity** Reconfiguration cannot operate on any random MS concept, there must be reconfiguration units. As mentioned in [7], modularity is a basic requirement for RMS and module-based configuration enables reconfiguration. *Indeed, elements can be put together either to adjust the production volume (scalability), or to add functionality to the system (convertibility), or to produce more customized products (customization).*

③ **Type and Instance Models** As mentioned in [2], in MES we need both type and instance at the modelling level because instances can have several occurrences. A resource-type (*e.g.* a arm robot) describe capabilities. A resource (*e.g.* the FANUC LR Mate 200) describe a resource instance that may have several occurrences (instances), each having its customised information (id, values...). The same applies for products defined by product-types (*e.g.* with a nomenclature) and produced in several instances according to a product order. In Object Orientation, meta-protocol allow the three-level instanciation: an instance is defined by a class which is defined by a metaclass (right part of Fig. 1). In our case we will simply use class modelling (right part of Fig. 1): a module-type defines module that are instantiated for each occurrence.

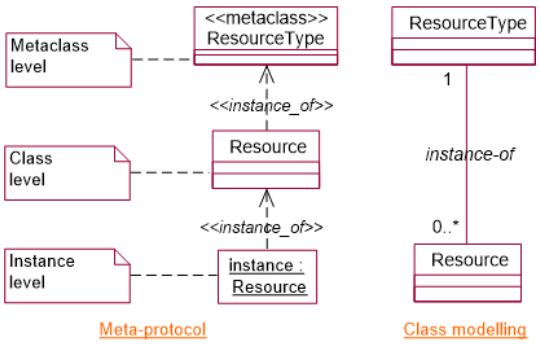

Fig. 1 Types vs instance models

④ **Abstraction and encapsulation** Abstraction is a key feature in modular design. A configuration is an assembly of modules that focuses on the module interface (an abstraction) not the module implementation (encapsulates data, control...). The interface must include the information to decide whether modules can be linked together or not *(integrability)*.

⑤ **Recursive aggregation** A module can be implemented by other modules. From the interface point of view, a composite is a module that can be assembled with other in a configuration. From the implementation point of view, a composite modules encapsulates a configuration of other modules. This is an elegant way to achieve *scalability* and improves part to part *diagnosibility*. For example, a module can be a single resource or a workstation module in a workshop or a production line module in a factory module.

⑥ **Multi-aspect and loose coupling** A manufacturing system involves several stakeholders that have different concerns: CPS engineers to connect physical modules, automation engineers for resources control, software engineers for communication networks and MES, business managers for KPI, etc. Both modules and configurations are perceived differently by stakeholders. For example, Fotso et al. propose a fractal vision of modules where four aspects are associated to modules (physical, control, simulation, KPI) [7]. However their (high-level) reference model is not applicable because the glue for integration (based on the production process) has not been studied and the viewpoints are not continuously fractal among recursive aggregation. Last to be reusable the aspects must be as independent as possible (*loose coupling*).

⑦ **Early verification and strong consistency The consistent and complete integration of the viewpoints is the most important challenge to tackle for RMS applicability**. In addition to model modularity, we need modelling language modularity by composing Domain Specific Languages to face heterogeneity. As an example, we separate in three axis: system structure (organisation) from its dynamic behaviour (or control) and its functional behaviour (or actions, computations). Smaller DSL improves *verifiability* and *diagnosibility* by focusing on specific properties (*strong consistency*) *e.g.* deadlock freeness is associated to control not to the structure or computations.

4 RMS Reference Model

In this section we provide excerpt of the reference architecture, that has been built upon the principles of Section 3. The models will be illustred with UML and expressive modelling notation (principle ①). Fig. 2 shows the general organisation of the reference architecture. The RMS_Modules describes modules and configurations including layouts and organisational units. The RMS_Control layer adds dynamic information to control modules. The RMS_Product layer describes the products. The RMS_Production layer adds functional information for the choregraphy and

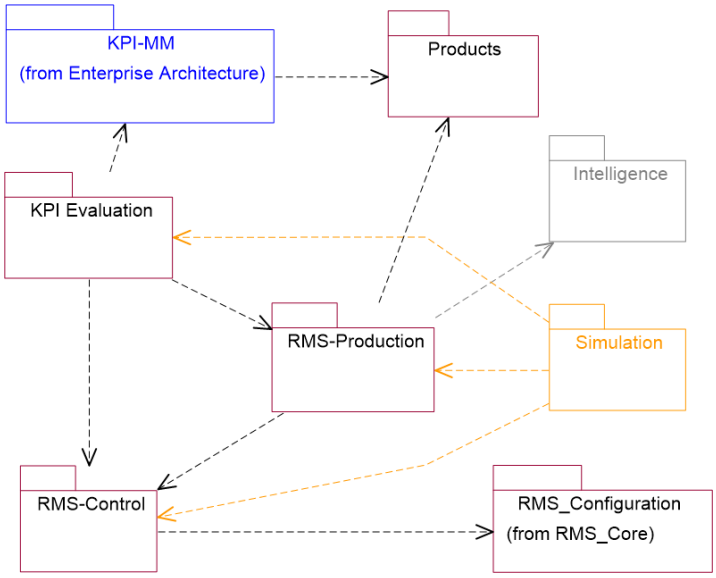

Fig. 2 Main Reference Architecture

orchestration of manufacturing. The Intelligence package adds facilities for scheduling and other ordering activities. The KPI–MM (metamodel) layer defines KPI, at the enterprise level basically on product information. The KPI are instrumented in the RMS by the KPI Evaluation integrating layer that provides evaluation means. Last the Simulation enables to compute metrics to evaluate the RMS before effective reconfiguration. Note that we are at a logical level, the physical part is hidden under RMS_Architecture and RMS_Control. Next we will focus on the main trends, Intelligence and Simulation are cross-cutting secundary concerns that won't be detailed here. A Simulation emulates the real RMS control to run production activity and feed the KPI Evaluation.

4.1 Modules and Configurations

The core concepts of the RMS reference architecture are represented in Fig. 3. A **module** defines information (attributes or data objects) and operations (principle ② and ④). The available operations (*e.g.* the functional capabilities of a resource) are published in Interface s. A Service is a high-level operation with an API (inner operations), service Contracts (QoS) and protocol (how to use the operations)[1]. EndPoints enable to connect modules through Bindings. An **assembly** is a set of modules connected by assemblyBindings on their end points: this is a client/server relation. For example, "*taking a product on a conveyor place*" can be seen as binding the "leave" operation of a conveyor module with the "take" operation of

[1] The distinction between service and operation may be arbitrary and somewhat confusing but this exists for expressiveness reasons: it enables a range of implementations from plain imperative programming to service oriented computing.

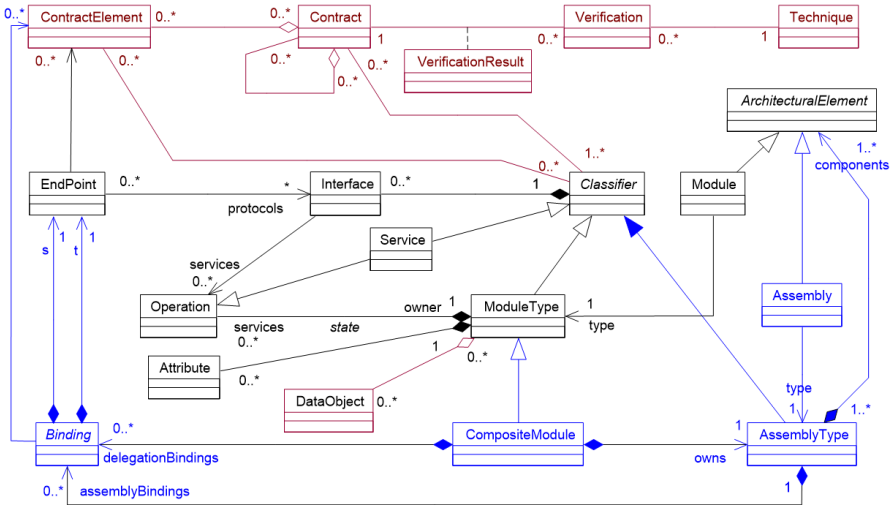

Fig. 3 Core RMS Configurations

a workstation module. However, operations as well as interfaces can be specialised as provided or required capabilities, this may depend on the implementation of the reference architecture. Behind the input parameters and output results, operations may exchange data through messages. Similarly to modules, AssemblyType is the abstraction of assemblies at the Module–type level (principle ③). CompositeModule enables scalability by encapsulating an assembly of modules (principle ⑤-recursion) and by being perceived as a (black-box) module of the assembly that contains it (principle ④-abstraction). The *orchestration* of composite services is realised via delegationBindings. Contracts are associated to end points and interfaces and enable the principle ⑦ (early verification). We won't detailed this point here but suggest the reader a previous work [1].

A **configuration** is basically an assembly of modules with initialisation information (the values for Attributes and Data objects). Note that configuration can be "implemented" by configuration services to control the consistency of the input parameters.

4.2 RMS-Control

Module as well as interfaces or operations (and services) are interacting actors and have a behaviour defined by dynamic expressions (*cf.* Fig. 4). For example, the behaviour of an operation is defined by a finite state machine (FSM) where the transitions are labelled by (atomic) actions or communication actions *e.g.* operation call, message send.... Since FSM denotes widespread formalisms, details are omitted here[2]. In this layer, we assumed a service-based communication with message send, signals are messages with no events. No matter what is the implementation of

[2] See the draft technical report - draft version`https://tinyurl.com/mumaaur3`

communications, having a service middleware enables to reason at a model level, for example to verify dynamic properties of contracts [12] according to principle ⑦. or to simulate the configurations (*e.g.* to compute KPIs).

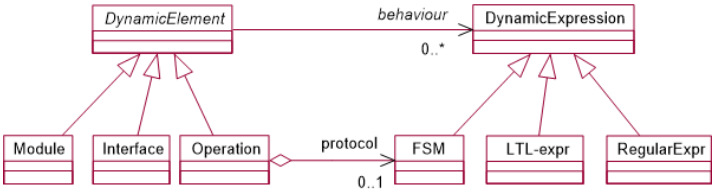

Fig. 4 RMS Dynamics

Several operations/services can run in parallel in a Module. The *choregraphy* is implicitly guided by operation and service calls but can be control by guarded actions (critical sections in concurrent processes). In CompositeModules this parallelism is implicitly distributed on the component modules but can be controlled by *orchestrating*, the delegationBindings through FSM protocols. Note that string encapsulation prevents component modules to collaborate with other modules than the other components modules of the same composite (assembly bindings) and the composite it-self (delegation bindings).

4.3 RMS-Production

The production layer includes every (abstract) concepts needed to organise the production process: products (and components, articles...), orders, storage, etc. We assume here a Products layer that defines products, which is a common topic to all MESs. We recall good modelling practice (recursive aggregation, type/instance separation, contract patterns...) can be found in [2]. The same reference gave examples of passive storage and containers, as well as passive orders and tasks information. Transportation and active storage are supposed to belong to RMS modules. Scheduling heuristics are assumed to be found in the Intelligence layer.

The RMS reference model set no constraints on the MES control organisation. Basically, in a centralised version of MES (orchestration), an order manager schedules the orders (and recursive sub-orders), each order is a sequence of tasks (recursive sequences of sub-tasks until reaching atomic tasks) (*cf.* Fig. 5). In a decentralised version of MES (choreography), an order manager is associated to every composite module to organise the production sub-process and enable partial reconfiguration or better runtime reconfiguration.

The MES dispatches orders on high-level modules according to product information (not explicit here). In [4], this coupling is designed by smart process/product decomposition of products, the process are bound to resources and the products are bound to orders. In [8] orders are specialised in Product-Order and Resource-Order each of them focusing on one side of product-order-ressource collaboration.

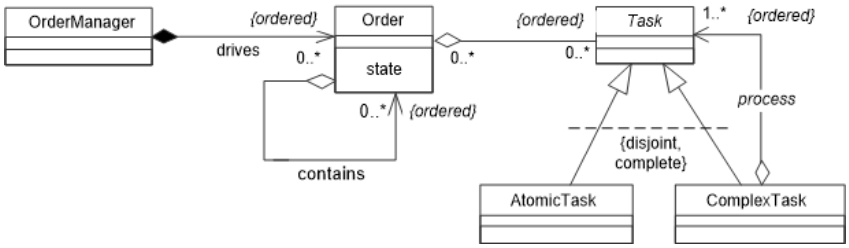

Fig. 5 Basic Order Management

The Order aggregation pattern of Fig. 5 enables finer coordination with product and resource holons. For example the tasks can be associated to resources only while the orders would be associated to products only. Note that orders and tasks can also be independent from resources and products to perform management or information processing.

4.4 KPI Computation

In many MES, KPIs are handled in a two standalone processes: the MES store the production events in databases and the KPI application queries the databases to compute KPIs by filtering on event types and the timestamps events under the software developer's responsibility. Reconfiguration may influence both the KPI definition and computation and if they are considered as a whole the entire KPI system is to be (costly) revisited. The idea in our reference model is improve modularity (and reuse) (principle ②) while separating the Enterprise and the Production concerns (principle ⑥). The KPI definition is KPI–MM is inspired from KPIML [13] that implements the ISO 22400 norm [9, 10]. It will always depend on the enterprise information system *e.g.* ERP and we assume here they focus on products (not the production resources). The main concern are first to define how to define explicitly aggregated KPIs of the management dashboard from production KPI and values (left part of Fig. 6) and second to feed KPI computations with values in respect with time periods (right part of Fig. 6). KPIs are defined according KPI–types (KPI definitions in KPIML) and the KPI computation tree is made explicit through aggregation where the expressions use the KPI codes. The tree leaves are computed on measures (metrics) on the production system by the means of Instruments: (1) Counter: inc/dec/raz - value, (2) Timer: start/stop/restart/reset - value and (3) Sensor: trigger - time stamped events.. These instruments are plugged on the RMS–Production over products, modules (and their resources), orders...

In this section we sketched the main concepts of the reference architecture. Next section will illustrate some of them.

5 Application on a Simplified Case Study

In this section, we provide the application of the approach to s simple case study.

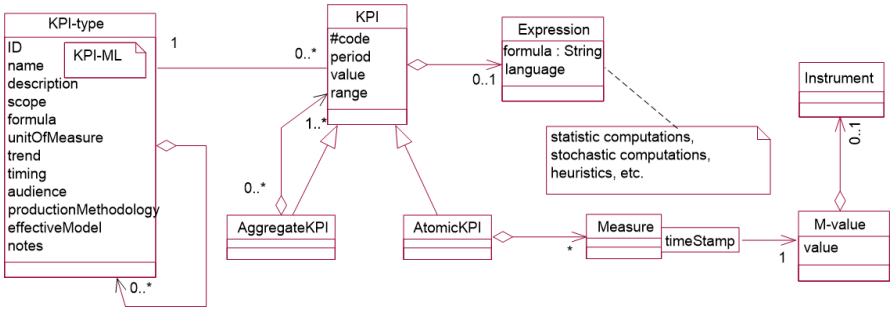

Fig. 6 RMS KPI

Case Study Outline

(Olivier) describe the case study, see reference Fig. 7

Model

The case study layout and module mapping are showed in Fig. 7. Every physical part of the system has to be represented somewhere in module model of Fig. 8.

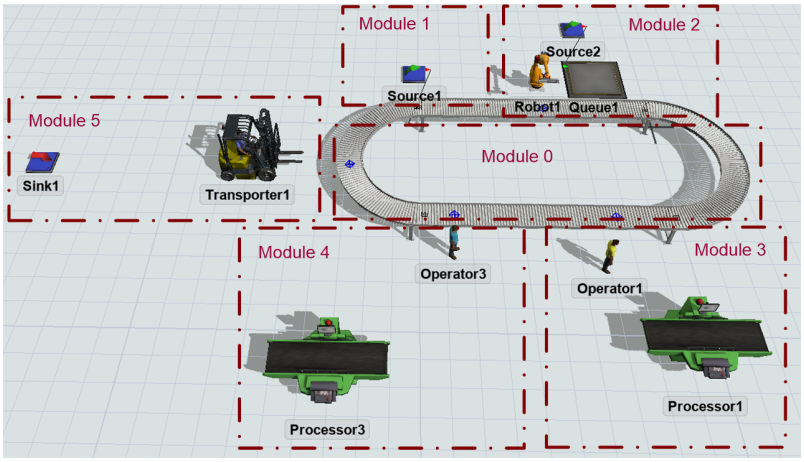

Fig. 7 Workshop physical and logical organisation

This model is represented using the UML component diagram notation, where rectangles are modules, square are end-points, lollipops are provided interfaces and semi-circles denote required interfaces. In this case study, the

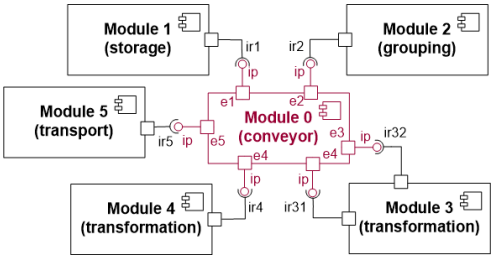

Fig. 8 Workshop modules model

modules are not in interaction directly but all are connected to the Module 0 (conveyor) that plays a role of bus communication architecture. Despite the end-point ep (pallet blocker) and interface of ip provided services (stopPallet , raiseUp, pick, put, rotate) are the same for all client module we represent it five times for readability reasons, however the corresponding interface ir_i of required services are different by default. Example of high-level services are: (1) Module 1 (addEmptyPallet), (2) Module 2 (addTwoProductsOnPallet), (3) Module 3 & 4 (takeProductFromPallet, processProduct, putProductOnPallet), (4) Module 5 (takeProductFromPallet, storeProduct), Of course running these services require interactions with Module 0 services. The fluid and energy flows are not defined here but they are considered as data and constraints in the interface and services. In Fig. 7, Modules 3 and 4 are clearly composite modules with two resources each, an operator and a processor. In the assembly of Fig. 8, only the module interfaces are available (black-box encapsulation). Note that Modules 3 has to end points access with Module 0 because Operator1 takes a product in one point and puts the product that has been transformed by Processor 1 in another place of the conveyor (Module 0).

The RMS–Control layer adds dynamic behavirours to modules. For example, the left part of Fig. 9 show the dynamic behaviour of service addTwoProductsOnPallet that loads two products from the store on the current pallet.The right part of Fig. 9 shows another possible behaviour that infinitely loads products on pallets.

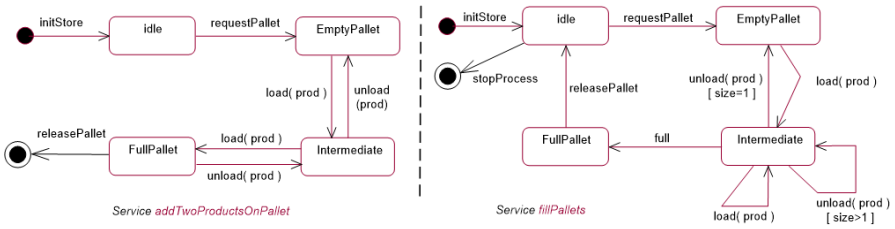

Fig. 9 Dynamic behaviour *addTwoProductsOnPallet*

The Products layer describes data objects: a single instance of ProductType that describes the operations to be performed on the Product instances that will travel between the modules. Each instance is identified by an product_id.

The KPI–MM includes the following performance indicators instances: (PA) OC: à revoir ensemble

- *OEE index = Availability * Performance rate * Finished goods ratio* ;
- Finished goods ratio - The finished goods ratio is the ratio of the good quantity produced (GQ) to the consumed material (CM). *Finished goods ratio = GQ / CM*
- The finished goods inventory shall be the amount of acceptable quantity which can be delivered. *FGI finished goods inventory*

- The quality ratio is the relationship between the good quantity (GQ) and the produced quantity (PQ).
- Throughput rate = PQ/AOET (Actual order Execution Time)
- Inventory turns *Inventory turns = TH / average inventory*.

In KPI Evaluation, we need metrics and equipments to compute the KPIs. For each measure that enters in AtomicKPI expression we provide instrumentation. The OEE index is instrumented by 3 timers for each module: waiting for product, waiting for pallet, production. The timers provide values for the availability and the performance. The AOET is implemented by one timer per order, starts when the production starts with the first product instance, stopped when the last product instance is manufactured. The consumed material (CM) value is instrumented by a counter associated to Module 5 that is incremented each time a product is stored in the inventory sink1 it is also the produced quantity (PQ). The good quantity (GQ) value is instrumented by a counter associated to Module 2 that is incremented each time a product is taken from the inventory source2. The number of processed products are stored in counters associated to modules 3, 4 and 5. The Inventory turns is implemented by a counter associated to Module 1 that is incremented each time a product is picked from the inventory source1. The counter and timers are time-stamped vectors when periodic rates are required (for time evolution feedback in simulation dashboards).

The Simulation part will define running scenarios and probabilistic events such as conveyor failure with possible reconfigurations with AGVs. The simulation aims to feed the KPI evaluation by providing values for counters and timers. As mentioned in Section 4, this part is out of the scope of the paper.

Next section discussed advanced topics.

6 Discussion

7 Conclusion

Conceptual models are missing to capture the the particularities of reconfigurable manufacturing systems (modularity, extendability, substituability, encapsulation...). In this paper we propose a reference model for RMS with the aim of being applicable by considering the different points of view on production systems. We detailed parts of these points of views be the means of packages and driving the models toward implementation. For example, the KPI vision is to be instrumented by a system observation instead of leaving this aspect to software providers. We illustrate on a simple case study how this is can be combined to describe a production system.

Actually we plan an implementation of this reference model for simulation purpose where the performance of each configuration is evaluated. This implementation should be later plugged to real systems according to a model-driven approach where a generic framework provides the execution engine [6]. Domain specific languages, dedicated to the points of views will be composed in order to preserve the consistency of each point of view (modelling simplicity, verification adequacy). Many design issues are to be solved including production task management, communica-

tion [3]. Such an implementation is the first and short-term perspective. In parallel, this reference model is the corner stone for reasoning on reconfigurations capabilities. Configurations are then considered as performance objects enriched by cost features and reconfigurations are operations on the objects ; costs are also associated to the operations. Such a system would enable to evaluate the balanced cost/profit reconfiguration.

Acknowledgments

Authors thank financial support from the French National Research Agency (ANR) under the RODIC project, grant number ANR-21-CE10-0017.

References

1. André, P., Cardin, O.: Trusted services for cyber manufacturing systems. In: T. Borangiu, D. Trentesaux, A. Thomas, O. Cardin (eds.) SOHOMA, pp. 359–370. Springer IP (2018)
2. André, P., Cardin, O.: Aggregation patterns in holonic manufacturing systems. In: T. Borangiu, D. Trentesaux, P. Leitão, O. Cardin, L. Joblot (eds.) Service Oriented, Holonic and Multi-agent Manufacturing Systems for Industry of the Future, pp. 3–15. Springer International Publishing, Cham (2022)
3. André, P., Cardin, O., Azzi, F.: Multi-protocol communication tool for virtualized cyber manufacturing systems. In: T. Borangiu, D. Trentesaux, P. Leitão, O. Cardin, S. Lamouri (eds.) Service Oriented, Holonic and Multi-Agent Manufacturing Systems for Industry of the Future - Proceedings of SOHOMA 2020, Paris, France, 1-2 October 2020, *Studies in Computational Intelligence*, vol. 952, pp. 385–397. Springer (2020)
4. Blanc, P., Demongodin, I., Castagna, P.: A holonic approach for manufacturing execution system design: An industrial application. Engineering Applications of Artificial Intelligence **21**(3), 315–330 (2008)
5. Bortolini, M., Galizia, F.G., Mora, C.: Reconfigurable manufacturing systems: Literature review and research trend. Journal of Manufacturing Systems **49**, 93–106 (2018)
6. El Amin Tebib, M., André, P., Cardin, O.: A model driven approach for automated generation of service-oriented holonic manufacturing systems. In: T. Borangiu, D. Trentesaux, A. Thomas, S. Cavalieri (eds.) SOHOMA, pp. 183–196. Springer IP (2019)
7. Fotsoh, E.C., Mebarki, N., Castagna, P., Berruet, P., Quintanilla, F.G.: Towards a reference model for configuration of reconfigurable manufacturing system (RMS). In: B. Lalic, V.D. Majstorovic, U. Marjanovic, G. von Cieminski, D. Romero (eds.) Advances in Production Management Systems. The Path to Digital Transformation and Innovation of Production Management Systems - IFIP WG 5.7 International Conference, APMS 2020, Novi Sad, Serbia, August 30 - September 3, 2020, Proceedings, Part I, *IFIP Advances in Information and Communication Technology*, vol. 591, pp. 391–398. Springer (2020)
8. Gamboa Quintanilla, F., Cardin, O., L'anton, A., Castagna, P.: A modeling framework for manufacturing services in service-oriented holonic manufacturing systems. Engineering Applications of Artificial Intelligence **55**, 26–36 (2016)
9. ISO: Iso22400: 1—automation systems and integration—key performance indicators (kpis) for manufacturing operations management—part. 1: Overview, concepts and terminology (2014)
10. ISO: Iso22400: 2—automation systems and integration—key performance indicators (kpis) for manufacturing operations management—part. 2: Definitions and descriptions (2014)
11. Kaiser, J., McFarlane, D., Hawkridge, G., André, P., Leitão, P.: A review of reference architectures for digital manufacturing: Classification, applicability and open issues. Computers in Industry **149**, 103,923 (2023)

12. Messabihi, M., André, P., Attiogbé, J.C.: Multilevel contracts for trusted components. In: J. Cámara, C. Canal, G. Salaün (eds.) Proceedings International Workshop on Component and Service Interoperability, WCSI 2010, Málaga, Spain, 29th June 2010, *EPTCS*, vol. 37, pp. 71–85 (2010)
13. Muhammad, U., Ferrer, B.R., Mohammed, W.M., Lastra, J.L.M.: An approach for implementing key performance indicators of a discrete manufacturing simulator based on the iso 22400 standard. In: 2018 IEEE Industrial Cyber-Physical Systems (ICPS), pp. 629–636 (2018)