An Engineering Framework for Service-Oriented Intelligent Manufacturing Systems

Abstract

Nowadays fully integrated enterprises are being replaced by business networks in which each participant provides others with specialized services. As a result, the Service Oriented Manufacturing Systems emerges. These systems are complex and hard to engineer. The main source of complexity is the number of different technologies, standards, functions, protocols, and execution environments that must be integrated in order to realize them. This paper proposes a framework and associated engineering approach for assisting the system developers of Service Oriented Manufacturing Systems. The approach combines Multi-agent system with Service Oriented Architectures for the development of intelligent automation control and execution of manufacturing systems.

Keywords: Service Oriented Intelligent Manufacturing Systems, Multi-agent system, Service Oriented Architectures, Software Engineering Method

1. Introduction

The rapidly changing needs and opportunities of todays global market require unprecedented levels of interoperability to integrate diverse information systems to share knowledge and collaborate among organizations. Fully integrated enterprises are being replaced by business networks in which each participant provides others with specialized services [1]. Almost no enterprise is able to accomplish all the production processes to offer a product or a product service system (PSS) independently. As a result, the Service Oriented Manufacturing Systems (SoMS) emerges.

A promising approach to develop SoMS is the integration of Multi-agent

system (MAS) and/or Holonic Manufacturing Systems (HMS)[1] with Service Oriented Architectures (SoA). MAS represents one of the most promising technological paradigms for the development of open, distributed, cooperative, and intelligent software systems. It has already been successfully applied to the field of Intelligent Manufacturing Systems (IMS) [4]. Moreover, the areas of SoA and MAS are getting closer and closer. In fact, in spite of both trying to deal with the same kind of environments formed by loose-coupled, flexible, persistent and distributed tasks [5], MAS and SoA present some important differences, namely in terms of autonomy and interoperability (see [6] for an in-depth study). Some researches state that the lacks in terms of interoperability exhibited by the MAS solutions in the manufacturing field can be overcome by combining SoA principles, and especially by using Web services technology [7]. An example of this fact is the new approach of Service Oriented Multi-agent Systems (SoMAS).

One of the critical aspects when developing SoMS is the complexity of the development process and the system itself. The system engineer needs support from software engineering methods and frameworks in order to manage the complexity originated by the number of different technologies, standards, functions, protocols, and execution environments that must be integrated in order to realize the SoMS. To the best of the authors' knowledge there is no complete engineering method in the specialized literature that can assist the system engineer during the whole life-cycle (requirements' specification, analysis, design, implementation, validation and verification, deployment, maintenance and operation) of a SoMS. Moreover, todays' changing market requirements force these systems to have some agility, reconfigurability and flexibility features in order to respond in a satisfactory and competitive way. The authors believe that MAS approaches can help to achieve all these features, and propose a service oriented engineering framework specifically tailored to deal with the development of IMS based on services and oriented to services. In this work it is presented a MAS based infrastructure for as-

[1]HMS is a paradigm that translates the concepts of living organisms and social organizations developed by Koestler [2] to the manufacturing world. A holon is an identifiable part of a system that has a unique identity, yet is made up of sub-ordinate parts and is in turn part of a larger whole. The holons can represent physical resources and logic entities, with intelligent and cooperative capabilities. In this paper we use agents and/or holons in order to refer to the intelligent software components that made up an Intelligent Manufacturing System. For a detailed comparison of the two concepts see [3].

sisting the system developers when developing Service Oriented Intelligent Manufacturing Systems (SoIMS).

The proposed approach is called ANEMONA-S + Thomas framework. The initial ideas of the framework were presented in [8], whereas in current paper an in-depth analysis of the details of the framework is presented, augmented with new guidelines and steps that complete the framework.The background on SoMS is presented in Section 2, together with a discussion on related works. The proposed framework is described in Section 3. In Section 4 the usefulness and completeness of the approach is evaluated. The conclusions and future works are analyzed in Section 5.

2. Cloud Manufacturing and Service Oriented Manufacturing

Cloud manufacturing is a computing and SoMS model developed from existing advanced manufacturing models and enterprise information technologies. Cloud computing, Internet of Things, virtualization and service-oriented technologies, and advanced computing technologies are the information technologies that support the realization of cloud manufacturing.

In a cloud manufacturing system, various manufacturing resources and abilities can be intelligently sensed and connected into the wider Internet by means of SoA principles, see Figure 1. In this way the manufacturing resources and abilities, from *Providers*, are virtualized and encapsulated into different manufacturing cloud services (*Mfg Services*) that can be accessed, invoked, deployed, and on-demand used by *Consumers*. An example of such approach is that SCADA and MES functions start being provided as services partially located in service clouds [9].

Service-based manufacturing network (or collaborative manufacturing virtual enterprises) is the manufacturing paradigm for the production of products and PSS[2]. In service-based manufacturing network, each enterprise focuses on core businesses, outsources non-core businesses (buy *Mfg Services*), and provides producer services (sell *Mfg Services*) for one another to achieve rapid innovation and improve efficiency. In this way, the integration of service and manufacturing has also changed the product pattern and not only

[2]A PSS is an integrated system in which the traditional functionality of a product is expanded by additional services. PSS shifts the focus to the usage of the product, that is, the customer does not pay for the possession of a product, but for the use of the product or for the functionality he receives [10].

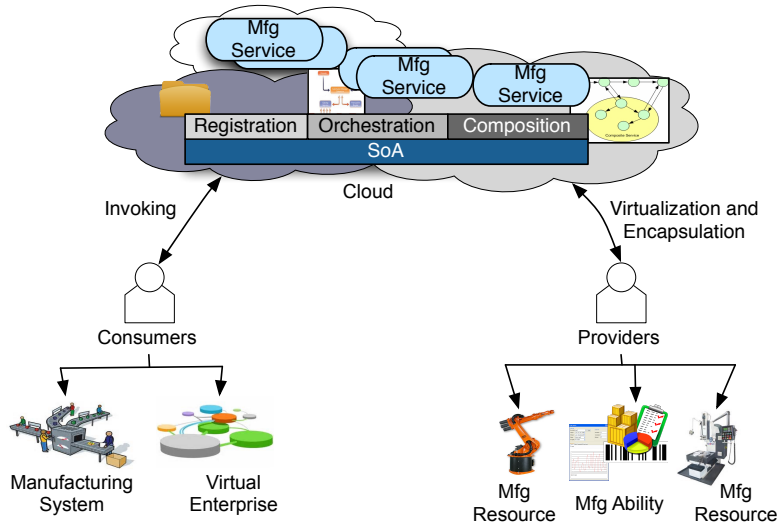

Figure 1: Cloud Manufacturing abstract execution

the manufacturing paradigm [11]. The physical product is servitized or integrated with services to form PSS.

The complexity of SoMS is hight due to large number of different elements and the need to integrate different components, standards, functions, protocols, and execution environments into a single system. Moreover, the development process of such system needs to be guided and supported by appropriate and complete software engineering methods that can help the developers to manage the complexity. Key fundamental features in the development process of SoMS are: (i) specific notation and method support for the identification and specification of the system components that will implement the *Providers* and *Consumers* in the SoMS; (ii) specific and complete guidelines to specify and implement the *Mfg Services* that will made up the SoMS; (iii) design and implementation artifacts that facilitate the development of agile, flexible and reconfigurable SoMS; (iv) an execution platform for supporting the deployment of SoMS in which open and dynamic system execution (run-time creation, modification and deletion of *Mfg Services*, *Providers* and *Consumers*) are allowed; (v) a uniform approach for design and implementation that facilitate the interconnection among components and levels in the SoMS (encouraging the use of standards). These key issues motivated the authors to propose ANEMONA-S + Thomas approach for the

development of SoMS.

In the specialized literature we can find some works in the field of SoMS. Adacor II [28] is an engineering framework based on distributed control systems and service oriented design paradigms. A Petri net-based language is used for intra and inter-coordination activities. Nevertheless, the proposed approach is very interesting and complete in terms of the set of tools that can be used, the approach lacks engineering guidelines. Moreover, there is a lack of uniform specification language and modeling concepts. In [29] an agent-based approach is described for modeling PSS. Nevertheless, the approach is limited only to the modeling of the life-cycle of the PSS after sell, i.e. in execution or in use. On the other hand [1] uses agent-based service-oriented approaches for the business level of virtual enterprise cooperation. In it the companies share resources by means of an agent-based infrastructure built on top of service-oriented technologies in order to define virtual enterprises. The SoMAS approach defined in ANEMONA-S is close to the work presented in [7]. Nevertheless, in [7] there is lack of methodological process to assist the system developer during system development. The ANEMONA-S method can be used to complement it in order to help the system designer at every development step. PROSIS [30] is a service oriented architecture based on PROSA type of holons as well as ANEMONA-S. PROSIS provides a set of specialized services that can be parameterized in order to use them in product, resource and order holons. Nevertheless, in PROSIS there is no methodological support for the development process. The SoA and MAS architecture described in [33] implements a set of layers and pre-defined platform supplied services that can be used for manufacturing automation level 2 of an ISA95/IEC 62264 standard architecture [31]. This proposal can be combined with the ANEMONA-S development process in order to populate the Service Cloud with application dependent services for specific manufacturing systems. The Thomas platform can be used also to implement the Service Cloud and the specialized services. IMC-AESOP [34] and SOCRADES [35] are two complete frameworks for implementing Cyber-Physical Systems using SoA. Nevertheless, both lack methodological support. ANEMONA-S development process can complement these two frameworks providing the support of engineering methods and notation.

Table 1 shows the list of abbreviation that are used in this paper.

Table 1: List of abbreviations

Abbreviation	Mining
ACL	Agent Communication Language
ASeM	Agent Supported e-Manufacturing environment
CMA	Customer Mediator Agent
FIPA	Foundation for Intelligent Physical Agents
HMS	Holonic Manufacturing System
IMS	Intelligent Manufacturing System
MAS	Multi-agent System
MES	Manufacturing Execution System
OMS	Organization Management Service
OVE	Open Virtual Environment
PSS	Product Service System
SCADA	Supervisory Control and Data Acquisition
SF	Service Facilitator
SoA	Service Oriented Architecture
SoIMS	Service Oriented Intelligent Manufacturing System
SoMAS	Service Oriented Multi-agent System
SoMS	Service Oriented Manufacturing System
SPEM	Software Process Engineering Metamodel
UML	Unified Modeling Language

3. ANEMONA-S + Thomas

ANEMONA-S + Thomas is a complete framework for developing SoIMS (see Figure 2). It is the integration of two main components: a MAS engineering methodology for SoIMS called ANEMONA-S; and the Thomas platform for the implementation and execution of open SoMAS. A system engineer can use them together and get the whole support from both components, or may opt by using just one of them and connect the development with other software tools or methods that are compatible with MAS and/or SoA.

ANEMONA-S is an extension of ANEMONA [12], which is a MAS methodology for HMS analysis and design. ANEMONA-S is based on service and oriented to service. Figure 2 shows the different components of the method. It provides complete support for SoIMS development, which includes: specific notation, complete and specific development process, tailored software engineering guidelines for SoIMS, CASE tool, implementation environment and Thomas service execution platform.

The SoIMS approach used in ANEMONA-S + Thomas framework is characterized by the use of a set of distributed autonomous and cooperative agents (embedded in smart control components) that use SoA principles. The agents are oriented by the offer (*Providers*) and request (*Consumers*) of services, in order to fulfill industrial and production systems goals. The manufacturing resources functionalities are encapsulated as services that can be offered,

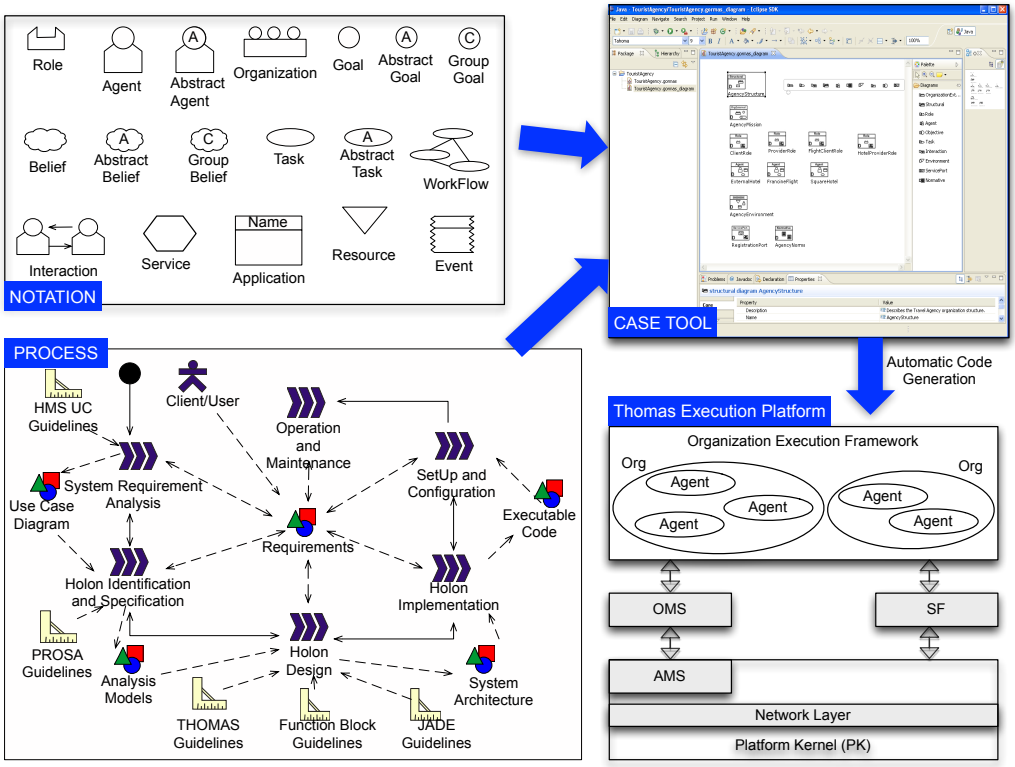

Figure 2: The components of ANEMONA-S + Thomas at a glance

searched and used by other entities, without knowing their underlining implementation. For this purpose, service providers publish the services they want to offer in a service registry (see Section 3.4), and the service requesters search the services they want to use through the help of discovery methods provided by Thomas [13].

3.1. Software Engineering Process

ANEMONA-S + Thomas provides a specific development process in order to: find out what services will implement a given SoIMS, how to specify the services, how the services will be orchestrated and choreographed to support the functionalities of the system, how to implement and execute the services, and how to maintain the system. It is an iterative and recursive process focused on service-oriented MAS. Figure 2 shows its development

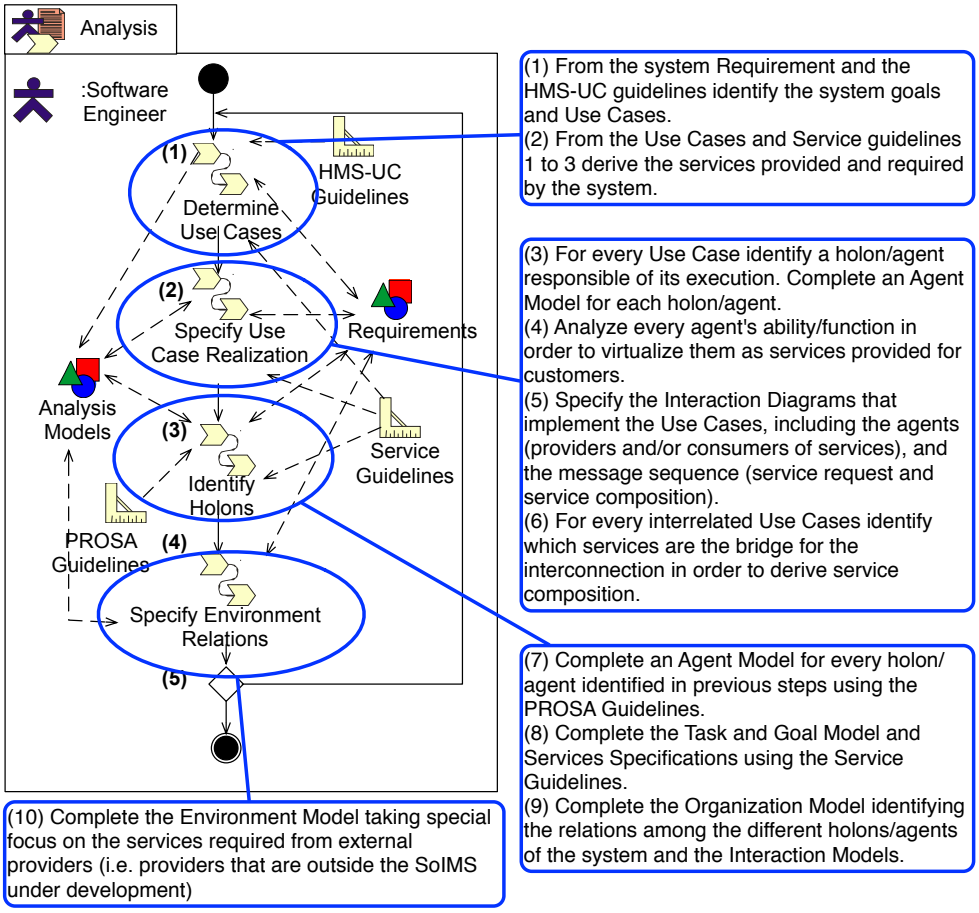

Figure 3: ANEMONA-S analysis activities

stages[3]. Every development stage has specific tasks that incrementally help the system designer to: figure out which system's functionalities can be virtualized and encapsulated as services provided for customers, which functionalities/services can be required/purchased from external providers, specify the services (including the business and physical level), specify and implement the providers and consumers of the services, deploy the system (including special support for local execution of services and services executed in cloud platforms).

[3]The specification of the development process is in SPEM (Software Process Engineering Metamodel) notation [14].

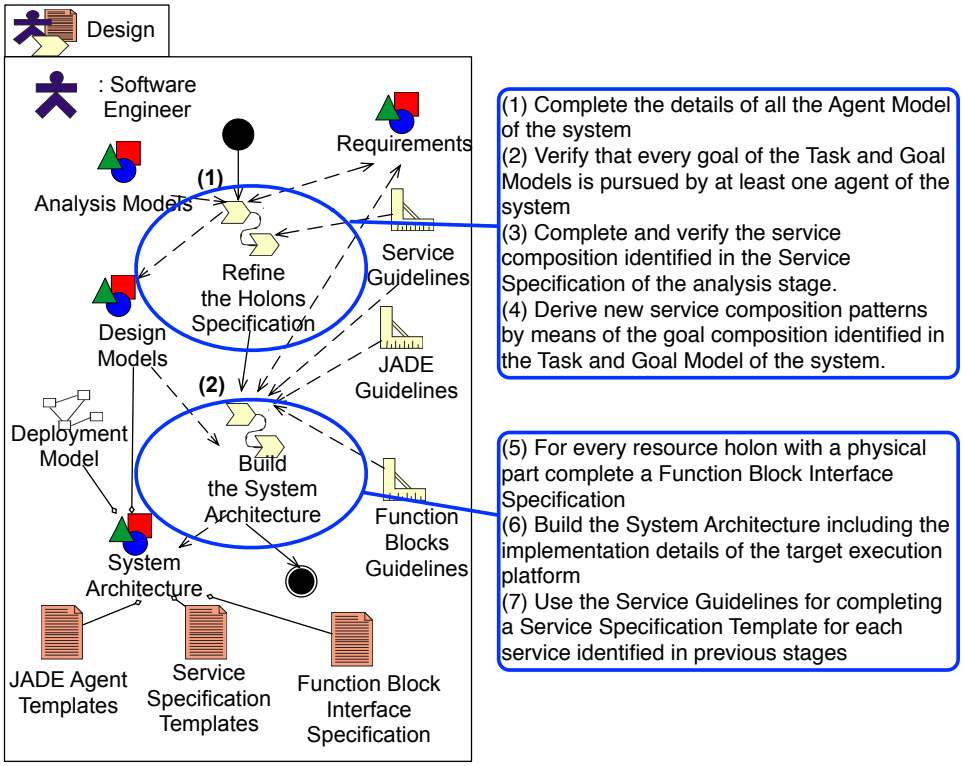

Figure 4: ANEMONA-S design activities

The analysis (Figure 3) adopts a top-down recursive approach and is supported by a set of specific guidelines (see Section 3.2) that help the system engineer to find out the agents that will provide and consume the services of the SoIMS. On the other hand, the design (Figure 4) is a bottom-up process to produce the system architecture from the analysis models. In this process the focus is on providing a complete specification of the services by means of the Service Specification Templates (see Section 3.3).

ANEMONA-S + Thomas uses SimIShop [15] simulation tool in order to evaluate the system correctness and get performance measures during the implementation and configuration steps. Moreover, an extension of Thomas is proposed as future work in order to connect it with the simulation tool based on NetLogo proposed in [16].

Table 2: ANEMONA-S Service Guidelines

ID	Guideline
1	Associate a new service description to each system functionality that is provided to external users of the system.
2	If the system requires a functionality from outside the system, identify a service request with it.
3	If two use cases requires interactions identify a new service to implement this interaction.
4	For every service (from guidelines 1 and 3) complete a *Service Specification Template* defined in [17]. Define the ontology for the service. To do this, analyze the system domain concepts. These concepts will be used to define the inputs, outputs and attributes of tasks, protocols and services.
5	For every service (from guidelines 1 and 3) make sure that there is one or more holons responsible for providing it.
6	Complete the holon specification from the service profile specified in the *Service Specification Template*. To define the service profile the attributes for each service must be identified. Create one activity diagram for specifying each service implementation. If there are services that should be published to other members of the system or to external stakeholders, the organizational diagram of the system should be refined by adding a service registry in the SF associated with the organization.
7	Complete the *Interaction Model* by means of sequences of services executions. Make sure that every service in an interaction model has its corresponding *Service Specification Template*.
8	Translate the *Service Specification Template* into Thomas code using the programers guidelines (http://www.gti-ia.upv.es/sma/tools/magentix2/index.php).

3.2. Service Oriented Guidelines

The guideline support provided by ANEMONA-S inherits from ANEMONA all the aspects for developing IMS. These guidelines are divided into six different groups (Figure 2), which are: HMS UC Guidelines [18] for identifying holonic manufacturing Use Cases from the requirements of the system; PROSA Guidelines [18] that help the system designer to identify and specify the agents/holons of the system in terms of PROSA [19] type of holons; Service Guidelines that help to virtualize and encapsulate manufacturing abilities of the agents/holons into services; Function Blocks Guidelines [18] for specifying the physical layer of the physical agents (e.g. machines, resources, etc.); and finally two sets depending on the chosen implementation platform, which are: THOMAS Guidelines [18] for Thomas development and execution platform [17], and JADE Guidelines [18] for JADE platform [20].

The set of Service Guidelines are exclusive of ANEMONA-S and specially

designed in order to follow a service oriented MAS engineering approach. Table 2 summarizes them. These guidelines are used at different development stages combined with the other set of guidelines provided by ANEMONA-S.

3.3. Agents and Services Specification

Agents and services are the two central entities in ANEMONA-S. The approach is that every agent's capability and/or ability is encapsulated into services. In this way the whole manufacturing system execution is implemented in terms of cooperations of different agents that try to fulfill the system goals by means of the execution of the services provided by agents and consumed by agents. The ANEMONA-S development process helps to identify the cooperation scenarios (or cooperations domains originally defined in Metamorph architecture [21]), the agents and the services. Moreover, the cooperation scenarios (Interaction Model, see previous section) guide the system developer to specify the patterns for service composition, choreography and orchestration. In the following paragraphs the key features that must be specified when modeling and implementing the agents are described. Together with the set of templates that complete the service specification.

ANEMONA-S supports PROSA types of holons [19]. An Agent Model is used to complete the definition of every agent/holon in the system. An agent is specified in terms of its goals, the ontology it understands, the cooperation scenarios in which it is involved, and the set of services it provides and/or consumes. Figure 5 shows the Agent Model of a Package Order agent of a Packing Cell System. This agent provides two services, i.e. Packing Cell Formation and Execute Packing, which are complex services orchestrated in terms of other services provided by a Packing Team of: Storage Agents, Conveyor Belts, Robots, Docking station, Wrapper agent and the Package Order agent.

The agents that have a physical part (a machine, tool, etc.) must have a Function Block Specification Interface (IEC 61499 - Function Block standard [22]) associated to complete their definition. Figure 6 shows the Function Block specification of a Conveyor Belt agent of the Packing Cell System.

The definition of product agents must state whether the product will be a PSS or not. In case it is a PSS it must also be specified if the service/s associated with it must be provided by the SoIMS or required from third parties (outsourcing non-core business processes).

A service entity is associated to an agent entity using two relations: provide and consume. A service entity is specified by means of two objects

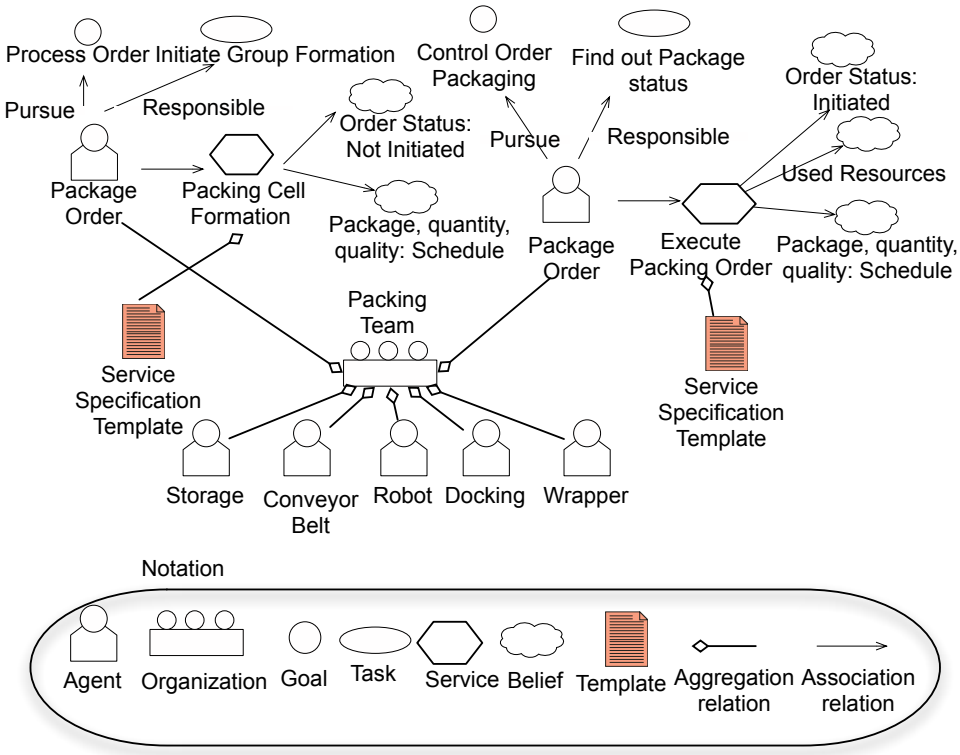

Figure 5: Agent Model of a Package Order Agent of a Packing Cell System

(compliant with SoA and Thomas specification [17]): the Service Profile represents the abstract service specification and is common to all providers; and a set of Service Process specifications, which details the concrete service implementation of each service provider. In the service specification there are included: a communication protocol, the message formats, the contact port, and other specific details of the service. It is specified using the OWL-S (Ontology Web Language for Services) standard (http://www.w3.org/TR/owl-guide/) extended with FIPA protocols. An ontology is associate also to give meaning to all of the elements of the service, using the OWL-DL language (http://www.w3.org/Submission/OWL-S/). At the same time a Service can be associated to a Workflow specification in order to define it with an approach similar to Business Process Execution Language and real time events. Finally, a service can be associated also to an Interaction entity in order to specify orchestration and composition. Table 3 shows the available services

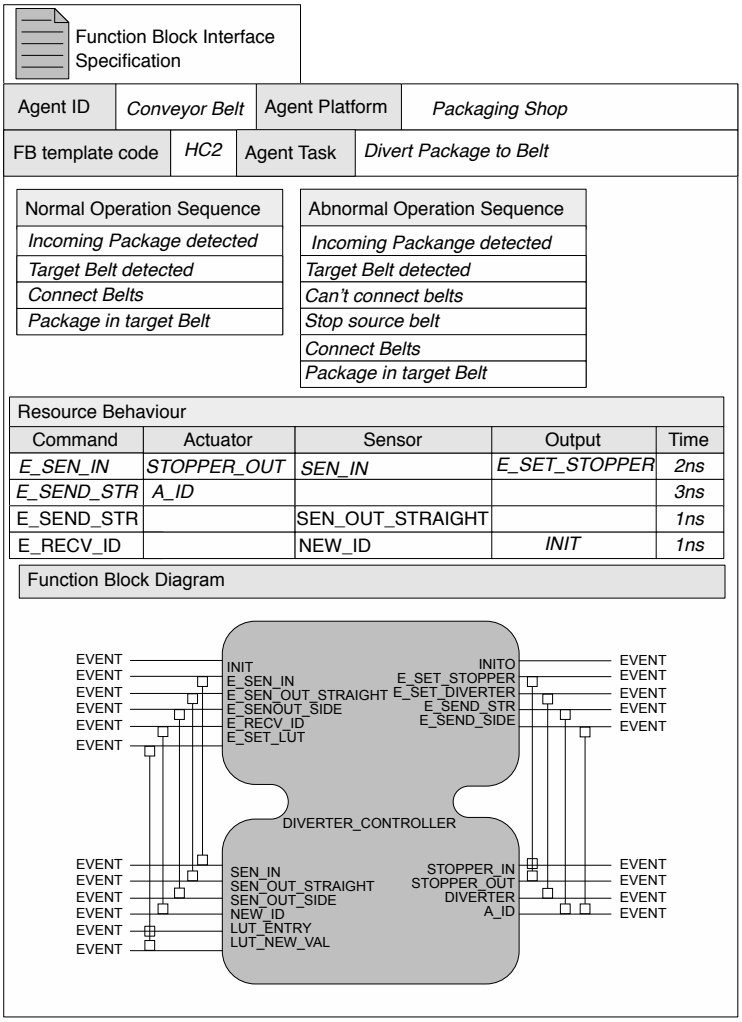

Figure 6: Function Block Specification Interface of the Conveyor Belt Agent of a Packing Cell System

provided by the Packing Team of the Packing Cell System. On the other hand Figure 7 depicts the Service Specification Template of the Execute Packing Order service.

The Service Specification Template is processed by ANEMONA-S + Thomas in order to generate an OWL-S description of the service in terms of: a unique service identifier; a list of providers; a service goal that can be achieved by executing the service, and; the Service Profile, which specifies what the ser-

Table 3: Services provided by the Packing Team in the Packing Cell system

Provider Entity	**Service**	**Inputs**	**Outputs**
Docking	Lock Shuttle, UnLockShuttle	ShuttleEvent, OrderEvent, FinishEvent, LockShuttle-Flag	LockShuttleFlag, Notification-Event, UnLock-ShuttleFlag
Robot	GetItempsOp, GetItems	NotificationEvent, Material Stock, List of Item Types, NotificationEvent, Material Stock, List of Item Types	FinishEvent
Package Order	GetOrder, SendOrder, Packing Cell Formation, Execute Packing	NotificationEvent, Order-Code, FinishEvent	List of Item Types, OrderCode, PackageCode
Storage	QueryCarriersAndStorage, QueryStorage	List of Item Types	MaterialStock
Wrapper	PaperWrapper, PackageWrapper	Type of Wrapper, Package Dimensions	WrapEvent

Service Specification Template					
Name:	Execute Packing Order				
Description:	A composed service for executing and controlling a packing order into the Packing Cell system				
Supplied by:	Package Order				
Required by:	External customers and Logistic Manager				
Input Parameters					
Name	**Description**	**Mand.**	**Type**	**Value Range**	**Default**
NotificationEvent	Is the activating event for starting the service	Yes	Enum	Start	
OrderCode	Unique ID of the packing order	Yes	String		
Output Parameters					
Name	**Description**	**Mand.**	**Type**	**Value Range**	
List of Item Types	Defines the list of items ID that must be packed in the order	Yes	Structure		
OrderCode	Unique ID of the packing order	Yes	String		
PackageCode	Unique ID of the package	Yes	String		
Pre-condition					
Pre1	$\exists\, PO, St, Cb, R, D, W \in PT \mid PO.role = PackageOrder \wedge St.role = Storage \wedge Cb.role = ConveyorBelt \wedge R.role = Robot \wedge D.role = Docking \wedge W.role = Wrapper$				
Post-condition					

Figure 7: Service Specification Template of the Execute Packing Order service of a Packing Cell System

vice fulfills, its constraints, the quality of service, and the requirements for the customers. On the other hand the Service Process is specified and associated to the concrete agent that provides the service. It includes the: service model in OWL-S, which specifies how the client can use the service, and; the service grounding (also in OWL-S), which specifies the communica-

tion protocol, the message formats, the service port, etc. During design and implementation steps (Figure 2) the service description is divided into two levels: business and physical. The business level is specified with OWL-S standard (as explained in previous paragraph), whereas the physical level with IEC 61499 [22]. Moreover, during the specification of the physical level an UML activity diagram as in [23] is used. In ANEMONA-S the two levels are connected by means of encapsulation and FIPA ACL messages. As in the proposal of Marariu et. al. [24] for a SoA implementation of an HMS, the resulting set of business process definitions, services, and schemas make up the logical architecture of the business level[4]. This logical architecture is mapped to a physical architecture relating the components of the application (agents) to a set of functional capabilities for the existing components of the SoIMS, including the tools, machines, devices, etc. This mapping into the physical architecture results into the System Architecture, which is a specification of the actual hardware, capacity, operating system, language, availability, policy and management system requirements to be actually used in the production system (the service providers).

Due to the complexity of manufacturing systems, there is an urgent need to use standards in the implementation and execution of these systems. ANEMONA-S + Thomas integrates the layers of ISA-95 [31] standard by means of agents that wrap the automation execution and control for manufacturing systems. Figure 8 depicts the relation of a SoIMS developed with ANEMONA-S + Thomas with the different layers of ISA'95. Every component/function that appears in ISA'95 can be virtualized as a service that is provided by an intelligent agent in ANEMONA-S + Thomas or is used as a service provided by others in the cloud and connected (by means of SoA architecture) with the agents in the framework. The different channels for intra-layer and inter-layer communication in ISA'95 are wrapped into agent messages (FIPA ACL standard) that are compatible with the Enterprise Service Bus model of SoA. The internal view of the logic architecture (Figure 8) is similar to the proposal of Colombo et. al. [32] for wrapping SCADA and MES functionalities into services.

[4]Nevertheless the proposed approach differs from [24] since it is not required a different implementation for every type of device due to the Thomas agents are created automatically with SoA compatibility. In this way it is possible to directly use an orchestrated architecture, based on Business Process Execution Language workflows and real time events

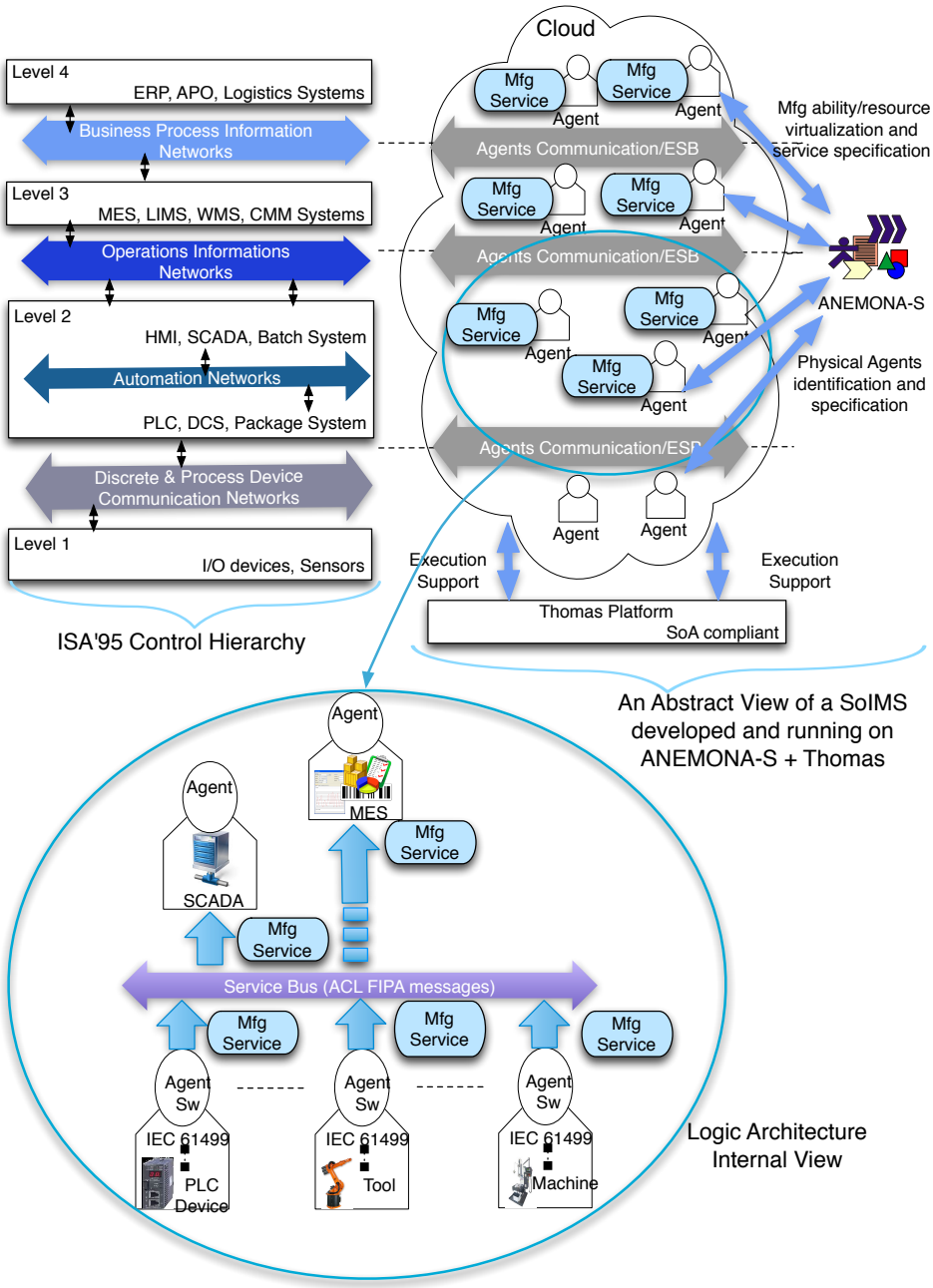

Figure 8: ISA'95 Control Hierarchy and its relation with an abstract view of a SoIMS running on ANEMONA-S + Thomas

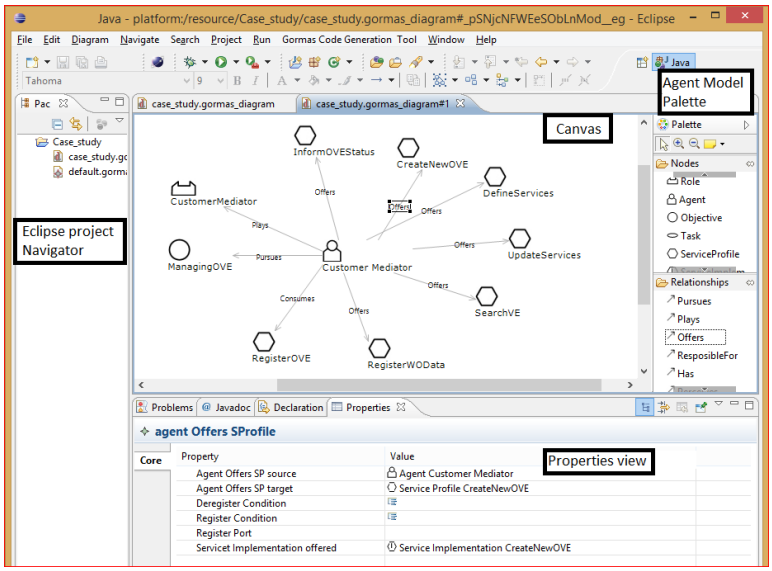

Figure 9: CASE tool snapshot

3.3.1. CASE Tool

One of the main components of the framework is the CASE tool. It is the software tool to design and implement SoIMS following the Model Driven Architecture standard and using the Eclipse technology. This tool provides a graphical editor for each of the models of the ANEMONA-S metamodel and integrates all models into a unique ecore file. Moreover, the graphical editor transforms these diagrams into Java skeletons that are ready to fill and execute in Thomas.

Figure 9 depicts a snapshot of the tool in which its main windows and functionalities are highlighted. The CASE Tool integrates the ANEMONA-S guidelines as a built-in help functionality for the system designer. Moreover, correctness checks are included in order to assure the coherence of the different models of the designed SoIMS.

3.4. Thomas

Thomas [17] is an open-agent platform that uses a service-based approach as the basic building blocks for creating SoMAS. Thomas aims at providing high quality software, which could be used in industrial applications. It

provides complete support for the management of virtual organizations for dynamic, open and large-scale environments. Figure 2 depicts its abstract architecture, which is based on the FIPA architecture (http://www.fipa.org), extending its main components. The FIPA Agent Management System and the Directory Facilitator are replaced by the Organization Management System (OMS) and the Service Facilitator (SF), respectively. The SF is a service manager that provides registration facilities for services provided by external entities and facilitates service discovery and orchestration for potential clients. On the other hand, the OMS is responsible for the management of virtual organizations, taking control of their underlying structure, the roles played by the agents inside the organization and the norms that rule the system behavior. In this way Thomas differs from well-known agent platforms, like JADE (http://jade.tilab.com), FIPA-OS (http://www.fipa.org), APRIL (http://www.nar.fujitsulabs.com/app/), among others, offering encapsulation of services and extended support for the management of virtual organizations. On the other hand, the main different with JADE WSIG (JADE - Web Service Integration Gateway, the extension of JADE for implementing SoMAS) [25] is that Thomas offers built-in services for managing service registration, discovery, composition, and orchestration, as well as built-in security and tracing services.

In Thomas the SF can be configured as a single service or as a federation of SFs. This is the main component that provides the SoA support. The registration, search, composition and orchestration of services are provided by the SF. It is implemented on an Apache Tomcat server (http://tomcat.apache.org) and executed as an internal platform service.

The Enterprise Service Bus is implemented by means of the communication layer of Thomas (Platform Kernel). In it the communication manager executes all the duties of the Enterprise Service Bus. Moreover, services can be used by the pull-principle, what can also be event-based. Thomas provides, among others, built-in security services and tracing tools. The Tracing Service Support allows the service clients to subscribe to an event associated to a service. If the subscribed event occurs, the registered client will be informed.

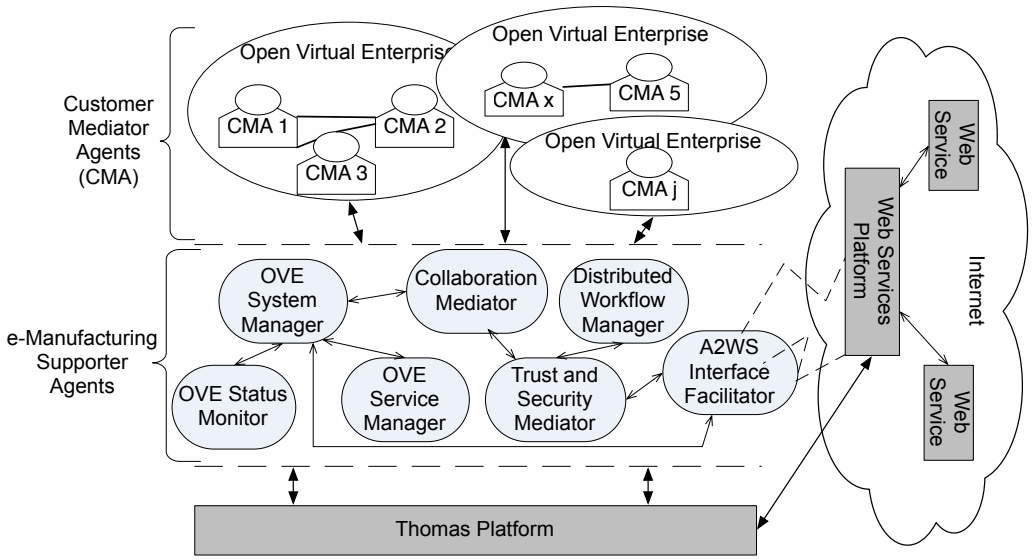

Figure 10: System Architecture of ASeM

4. Evaluation and Case-study

4.1. Case-study: an e-Manufacturing environment

In order to evaluate the proposed framework we executed an experiment in which two different engineering teams, with the same expertise on agent-based technology, developed a Case-study using ANEMONA, and ANEMONA-S + Thomas respectively. The goal of the evaluation was to figure out the easy to use and usefulness of the Service Guidelines, and the completeness of the proposed framework. The Case-study is an Agent Supported e-Manufacturing environment (ASeM). ASeM is a Web-based application in which different companies can take part into open virtual enterprises (temporary cooperations between various partners and services to support a set of activities).

Figure 10 shows the agents of ASeM. The lower level agents represent the e-Manufacturing Supporter Agents, which are local agents that provide ASeM environment services for the execution of the Customer Mediator Agents (CMAs). The OVE System Manager manages the different virtual enterprises in ASeM. The OVE Status Monitor keeps track of the status of running virtual enterprises. The OVE Service Manager is responsible for managing the service directory in ASeM. The Collaborator Mediator agent

is responsible for mediating among CMAs interacting in collaboration scenarios. It facilitates the service and collaboration scenario discovery among the registered users. The service provision rule of Thomas is maintained, i.e. a first-in-first-out policy among the clients that are allowed to request the service and have requested it. The Trust and Security Mediator maintains trust and security data of the CMAs. The Distributed Workflow Manager keeps track of the status of the work orders that are being processed in the different virtual enterprises running in ASeM. The A2WS Interface Facilitator provides the set of services required for a Web-based presentation layer (JSON[5] - JavaScript Object Notation is used for the dynamic creation of the web-pages). A CMA keeps record of a manufacturing company description, together with collaboration related data such as the set of services the company offers and its execution log. CMA provides a complete set of services to its associated company: agent configuration for service updates, search service for joining virtual enterprises , data management service, virtual enterprise status service, agreements query service, work-orders management service, virtual enterprise exit service, etc. Figure 11 shows the CMA agent model in which these services are specified. Whereas Figure 12 depicts the cooperation diagram, with the steps and FIPA ACL messages, for a virtual enterprise creation. In this particular sequence diagram it can be noticed the interaction of the CMA with a group of supporter agents in order to create a new virtual enterprise. In virtual enterprises scenarios security and trust information is key. In order to support it the Trust and Security Mediator keeps record of the potential customers of any virtual enterprise. This agent maintains a trust and reputation model of the agents, based on the approach defined in [26], in order to get a ranked list on the customers and their interaction in the system. In this way the OVE System Manager can create and activate the virtual enterprise only if the requester CMA surpasses a given trust and reputation value threshold.

Table 4, shows the overall results of the experiments with the two development teams. The first team derived a system with zero service (the functionalities of ASeM are provided as conventional agents capabilities). Whereas, the second team derived a system implemented by a service-oriented multiagent system. This is the main reason for the results on the last column about failed test to add/delete virtual enterprises. The user interaction with

[5]http://www.json.org

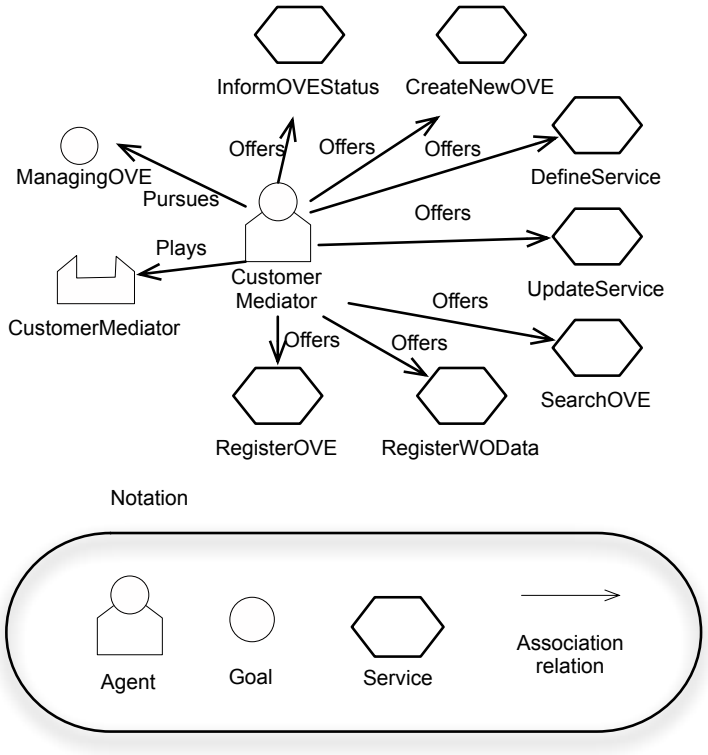

Figure 11: CMA agent model

the second system was much more easy and very "smooth" for adding and deleting virtual enterprises during run-time. In terms of functionality and number of holons, the second team derived a system with fewer holons (9 vs 13) that compiles similar services into fewer providers. In terms of development time, the first team finished first the system. This could be because no team was expert on SoA implementation and there was required two training weeks for the second team in order to get the services running in the platform. Finally, the second team required only 2 iterations of the development process to complete the system development thanks to the Service Guidelines. The second team highlighted the usefulness of the built-in help provided by the CASE tool and the automatic code generation of service profiles and processes. These two features helped to reduce the implementation and configuration time. Nevertheless, for debugging activities both teams agreed that the debugging tool provided by Thomas is not easy to use nor

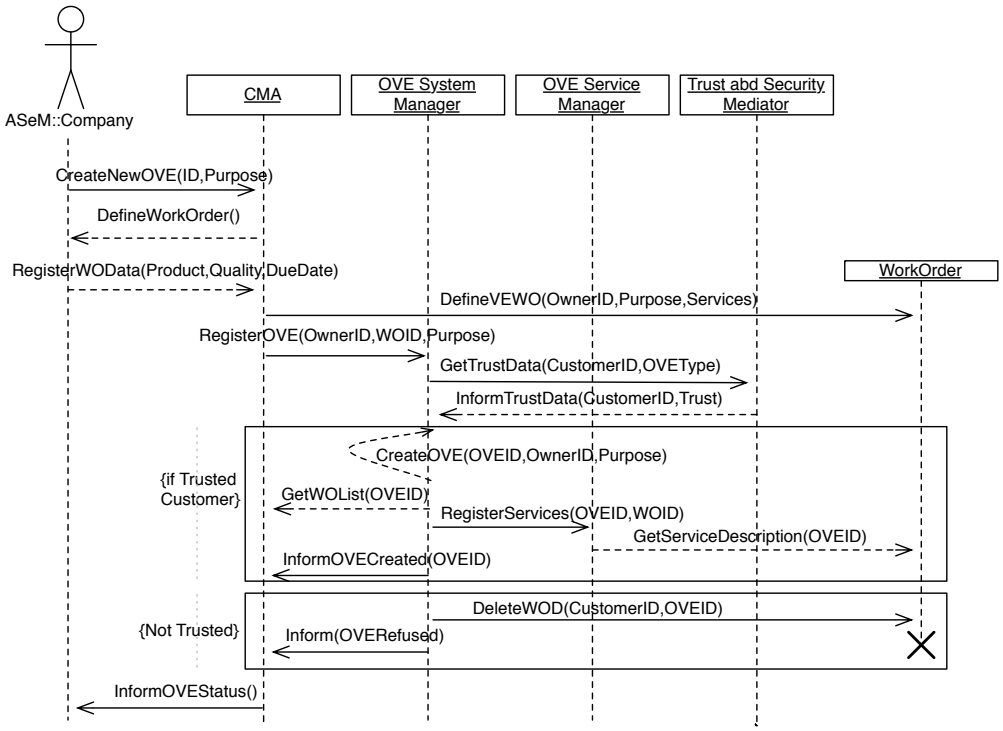

Figure 12: Cooperation diagram for virtual enterprise creation

configurable (this is an aspect already identified in previous developments and is being re-designed in order to provide a better tool). It is also important to point out that the second team recommended to add a CASE tool module with pre-built interaction patterns for: team formation, monitoring and deletion for different typical manufacturing scenarios, such as production line, maintenance, supply chain, warehousing, delivery, etc.

The second team was asked about how easy and/or hard is to combine services with agents using ANEMONA-S + Thomas. All the members of the team agreed that the Service Specification Templates were easy and, sometimes, straightforward to complete from the Agent Model and Interaction Model. Moreover, the automatic code generation of Thomas for service profile and process reduces the implementation work to an activity of adjusting and completing minor details. In conclusion the second team stated that at the end combining services with agents was not hard, and that SoMAS

Table 4: Case-study experiment results for two teams of 3 members each

Team	**Service bkgd**	**Time to finish**	**Training Time**	**Total Agents**	**Total Services**	**Total Iterations**	**Failed Tests**
ANEMONA	No	2,5 month	0	13	0	5	23/100
ANEMONA-S + Thomas	No	3,2 month	0,5 month	9	12	2	2/100

provided more flexibility to the system they developed.

4.2. Qualitative Evaluation

In order to analyze and compare ANEMONA-S + Thomas with other approaches we have used a MAS evaluation framework called MASEV [27]. This framework allows analyzing and comparing methods and tools for developing MAS in terms of general requirements, offered method guidelines and the support for the integration of MAS and services. After this analysis[6] we can conclude that ANEMONA-S + Thomas offers an extensive support for integrating services into SoIMS (outperforming 5 reviewed methods). Also we have observed that ANEMONA-S + Thomas differs from other proposals by providing specific guidelines for the manufacturing domain (outperforming 15 reviewed methods).

5. Conclusions

In this work it is proposed a specific framework for developing SoIMS. The proposed framework is made up of a new methodology called ANEMONA-S and the Thomas platform. The proposed framework provides complete support for SoIMS development and execution. The proposed approach was validated by means of an experiment in which a Web-based application for virtual enterprise definition and execution was implemented. The results of the experiment show that the framework help to develop a correct and complete system. Moreover, the specific guidelines for SoIMS help the system developer to execute less process iterations and to identify fewer agents that implement the same number of system functions. In the qualitative evaluation the framework was analyzed and compared with other approaches using a MAS evaluation tool. From this evaluation the framework outperformed

[6]The details of the analysis are available at http://masev.gti-ia.dsic.upv.es

others approaches in terms of the offered support for identifying and describing services and also on the service specific guidelines support for SoIMS.

As future works there are two lines identified: evaluation and enhancement/addition. About evaluation, more work is required to evaluate and validate the framework in real industrial environment having real industrial manufacturing conditions in order to assure a given quality and readiness level to the industry. On the other hand, it has been identified a set of enhancement/addition in order to be able to connect the framework with other tools and environments. Such as the tool proposed in [16], the approach presented in [7] and the architecture described in [33]. IMC-AESOP [34] and SOCRADES [35] are two promising architectures for cloud-based industrial Cyber-Physical Systems. Analyzing the feasibility and completeness of the models of ANEMONA-S for developing applications in these two architectures is also foreseen as an interesting future work.

[1] W. Shen, Q. Hao, S. Wang, Y. Li, H. Ghenniwa, An agent-based service-oriented integration architecture for collaborative intelligent manufacturing, Robotics and Computer-Integrated Manufacturing 23 (3) (2007) pp. 315–325.

[2] A. Koestler, The Ghost in the Machine, Arkana Books, London, (1971).

[3] A. Giret, V. Botti, Holons and Agents, Journal of Intelligent Manufacturing 15 (2004) pp. 645–659.

[4] P. Leitao, V. Marik, P. Vrba, Past, Present, and Future of Industrial Agent Applications, IEEE Transactions on Industrial Informatics 9 (4) (2013) pp. 2360–2372.

[5] M. Huhns, M. Singh, et.al., Research directions for service-oriented multiagent systems, IEEE Internet Computing. 9 (1) (2005) pp. 52–58.

[6] L. Ribeiro, J. Barata, P. Mendes, MAS and SOA: Complementary Automation Paradigms., in: Azevedo, A. (ed.) Innovation in Manufacturing Networks. IFIP, vol. 266, Boston, (2008) pp. 259–268.

[7] P. Leitao, Towards Self-organized Service-Oriented Multi-agent Systems, in: T. Borangiu et al. (Eds.): Service Orientation in Holonic and Multi Agent, SCI 472, Berling, (2013) pp. 41–56.

[8] A. Giret, V. Botti, ANEMONA-S + Thomas: A Framework for Developing Service-Oriented Intelligent Manufacturing Systems, in: T. Borangiu et al. (eds.), Service Orientation in Holonic and Multi-agent Manufacturing, Studies in Computational Intelligence 594, Berling, (2015) pp. 61–70.

[9] S. Karnouskos, A. W. Colombo, Architecting the next generation of service-based SCADA/DCS system of systems, in: 37th Annual Conference of the IEEE Industrial Electronics Society (IECON 2011), IEEE Press, Melbourne, Australia, (2011) pp. 359–364.

[10] T. S. Baines, H. W. Lightfoot, S. Evans, et.al., State-of-the-art in product-service systems, Journal of Engineering Manufacture 221 (10) (2007) pp. 1543–1552.

[11] J. Gao, Y. Yao, V. Zhu, L. Sun, L. Lin, Service-oriented manufacturing: a new product pattern and manufacturing paradigm, Journal of Intelligent Manufactruing 22 (2011) pp. 435–446.

[12] A. Giret, V. Botti, Engineering Holonic Manufacturing Systems, Computers in Industry 60 (6) (2009) pp. 428–440.

[13] D. V. E., Service discovery in Open Service-Oriented Multi-Agent Systems, AI Communication 27 (2014) pp. 291–292.

[14] O. M. G. OMG, Software Process Engineering Metamodel Specification Version 1.0, http://www.omg.org/docs/formal/02-11-14.pdf.

[15] N. Ruiz, A. Giret, V. Botti, V. Feria, An Intelligent Simulation Environment for Manufacturing Systems, Computers & Industrial Engineering 76 (2014) pp. 148–168.

[16] C. Morariu, O. Morariu, T. Borangiu, Modeling and Simulation for Service-oriented Agent Based Manufacturing Systems, in: 2012 IEEE International Conference on Automation Quality and Testing Robotics (AQTR), IEEE, (2012) pp. 44–49.

[17] E. Argente, V. Botti, C. Carrascosa, A. Giret, V. Julian, M. Rebollo, An Abstract Architecture for Virtual Organizations: The THOMAS approach, Knowledge and Information Systems 29 (2) (2011) pp. 379–403.

[18] V. Botti, A. Giret, ANEMONA: A Multi-agent Methodology for Holonic Manufacturing Systems, Springer. Springer Series in Advanced Manufacturing, (2008).

[19] H. Van Brussel, J. Wyns, P. Valckenaers, L. Bongaerts, P. Peeters, Reference Architecture for Holonic Manufacturing Systems: PROSA, Computers In Industry 37 (1998) pp. 255–274.

[20] JADE, http://jade.tilab.com/.

[21] F. Maturana, D. Norrie, Multi-Agent Mediator Architecture for Distributed manufacturing, Journal of Intelligent Manufacturing 7 (1996) pp. 257–270.

[22] IEC, International Electrotechnical Commission: Function Blocks, Part 1 - Architecture. PAS 61499-1, (2000).

[23] C. Legat, B. Vogel-Heuser, An Orchestration Engine for Services-Oriented Field Level Automation Software, in: T. Borangiu et al. (eds.), Service Orientation in Holonic and Multi-agent Manufacturing, Studies in Computational Intelligence 594, Berling, (2015) pp. 71–80.

[24] C. Morariu, O. Morariu, T. Borangiu, Customer order management in service oriented holonic manufacturing, Computers in Industry 64 (2013) pp. 1061–1072.

[25] F. Bellifemine, G. Caire, D. Greenwood, The JADE Web Services Integration Gateway, in: Developing Multi-Agent Systems with JADE, John Wiley & Sons, Chichester, (2007) pp. 181–205.

[26] A. Koster, J. Sabater-Mir, M. Schorlemmer, Opening the black box of trust: Reasoning about trust models in a bdi agent, Journal of Logic and Computation (JLC) 23 (1) (2013) pp. 25–58.

[27] E. Garcia, A. Giret, V. Botti, Evaluating Software Engineering Techniques for Developing Complex Systems with Multiagent Approaches, Information and Software Technology 53 (2011) pp. 494–506.

[28] P. Leitao, J. M. Mendes, A. Bepperling, D. Cachapa, A. Colombo, F. Restivo, Integration of virtual and real environments for engineering service-oriented manufacturing systems, Journal of Intelligent Manufacturing 23 (2012) pp. 2551–2563.

[29] S. Maisenbacher, D. Weidmann, D. Kasperek, M. Omer, Applicability of Agent-Based Modeling for Supporting Product-Service System Development, Procedia CIRP 16 (2014) pp. 356–361.

[30] Y. Dubromelle, F. Ounnar, P. Pujo, Service oriented architecture for holonic isoarchic and multicriteria control, in: T. Borangiu, A. Thomas, D. Trentesaux (Eds.), Service Orientation in Holonic and Multi-Agent Manufacturing Control. SOHOMA 2011, Springer-Verlag, Berlin, (2012) pp. 155–168.

[31] WBF, THE WBF BOOK SERIES-APPLYING ISA 95. Implementation Experiences. Wbf Book Series, World Batch Forum, Wbf, Momentum Press, (2010).

[32] A. Colombo, J. Mendes, P. Leitao, S. Karnouskos, Service-oriented SCADA and MES Supporting Petri nets based Orchestrated Automation Systems, in: 38th Annual Conference of IEEE Industrial Electronics (EICON 2012), (2012) pp. 6144–6150.

[33] K. Nagorny, A. Colombo, U. Schmidtmann, A service- and multi-agent-oriented manufacturing automation architecture: An IEC 62264 level 2 compliant implementation, Computers in Industry 63 (2012) pp. 813–823.

[34] S. Karnouskos, A. Colombo, T. Bangemann, K. Manninen, R. Camp, M. Tilly, M. Sikora, F. Jammes, J. Delsing, J. Eliasson, P. Nappey, J. Hu, M. Graf, The imc-aesop architecture for cloud-based industrial cyber-physical systems, in: A. W. Colombo, T. Bangemann, S. Karnouskos, J. Delsing, P. Stluka, R. Harrison, F. Jammes, J. L. Lastra (Eds.), Industrial Cloud-Based Cyber-Physical Systems, Springer International Publishing, (2014) pp. 49–88.

[35] L. de Souza, P. Spiess, D. Guinard, M. Kohler, S. Karnouskos, D. Savio, Socrades: A web service based shop floor integration infrastructure, in: C. Floerkemeier, M. Langheinrich, E. Fleisch, F. Mattern, S. Sarma (Eds.), The Internet of Things, Vol. 4952 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, (2008) pp. 50–67.