Adaptive Production Management Using a Service-Based Platform

Usman Wajid[1(✉)], Vadim Chepegin[2], Despina T. Meridou[3], Maria-Eleftheria Ch. Papadopoulou[3], and José Barbosa[4]

[1] The University of Manchester, Oxford Road, Manchester M15 6PB, UK
usman.wajid@manchester.ac.uk
[2] TIE Kinetix, De Corridor 5d, 3621 ZA Breukelen, Netherlands
vadim.chepegin@tiekinetix.com
[3] School of Electrical and Computer Engineering, National Technical University of Athens, Heroon Polytechniou 9, 15773 Athens, Greece
{dmeridou,marelpap}@icbnet.ece.ntua.gr
[4] Polytechnic Institute of Bragança, Apartado 1134, 5301-857 Bragança, Portugal
jbarbosa@ipb.pt

Abstract. This paper presents a platform for adaptive production management developed in the ARUM[1] (Adaptive pRodUct Management, http://arum-project.eu/) project. The design of ARUM platform started with applying a traditional enterprise Service-Oriented Architecture (SOA) paradigm to solving an integration problem for the production ramp-up of highly customized products such as aircrafts, ships, etc. The production of such articles is exceptionally challenging for planning and control, especially in small lot sizes. Often requests for changes at any stage of the production, immature products and processes bring serious additional risks for the producers and customers. To counter such issues requires new strategies, the core elements of most of them include early detection of unexpected situations followed by rapid mitigation actions. Furthermore, human beings cannot cope any longer with processing a massive volume of data that comes with a high velocity from various sources that is a requirement for any modern production shop floor. The traditional IT solutions also fall short when trying to satisfy all those requirements and this motivates the need for ARUM platform to help in effective decision making.

Keywords: System architecture · Adaptive manufacturing · Enterprise service bus

1 Introduction

Automation of conventional processes and wider adoption of service oriented system design approaches mean services are playing an increasingly important role in modern

[1] This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 314056.

V. Mařík et al. (Eds.): HoloMAS 2015, LNAI 9266, pp. 133–144, 2015.
DOI: 10.1007/978-3-319-22867-9_12

manufacturing domain. Currently there are two trends for designing service-based systems competing in the market that many researchers and practitioners consider mutually exclusive, namely service-oriented architecture (SOA) and REST style. To design a service-based system that brings together and integrates different entities SOA advocates the use of an Enterprise Service Bus (ESB) for integration pattern and functional decomposition of system level entitiesogies. However, at enterprise level the implementation of SOA is usually associated with large investments in IT infrastructures in terms of time and funds. On the contrary, REST integration style is capitalizing on the existing Web infrastructure. It supports P2P communications, lightweight protocols and message formats for interaction between different services, and does not require large upfront investments. The success and rapid growth of the Internet is often seen as a proof of success for the REST style. REST services are globally used for delivering new business functions via exploiting existing polyglot infrastructure (execution environments, persistency) and continuous delivery approach.

This paper describes a service-based platform that enables the manufacturing industry to benefit from using both of the paradigms together taking an example based on the experience and lessons learnt from the European funded ARUM (Adaptive pRodUct Management) project. ARUM aims at solving hard planning problems during a production ramp-up of complex and highly customised products such as new aircrafts with help of Multi-Agent Systems (MAS). The ARUM approach is based on the combination of innovative lines of commercial SOA-based products such as *TIE Smart Bridge* (TSB), a lightweight message and service bus that strongly supports SOA principles, along with *TIE Smart Integrator* (TSI) and a semantic mapping and transformation tool. TSB and TSI together provide facilities for data exchange and integration from a vast variety of services and legacy systems that cannot be reached using REST/HTTP on its own. Although, businesses in the manufacturing domain are typically dealing with common industrial formats and systems, within the ARUM context in order to satisfy the needs of knowledge-based MAS; TSI was enriched employing micro-services approach to take care of transforming data from legacy systems into the Web ontological models [2]. Both of the tools (TSI and TSB) can be exposed and be used via REST APIs, TSB has a support for workflows, and is designed to operate with SOAP and REST Web services, and altogether this solution paves the link between two integration paradigms, SOA and REST [4]. Moreover, TSB and TSI can be provided within the Cloud as Software-as-a-Service (SaaS) to help in integrating business services dispersed all over the Internet running on their own distributed Virtual Machines (VM) and Virtual Environments.

Thus, the proposed ARUM solution speeds up and facilitates delivery of the new value for the businesses by quickly setting up a foundation for communications in heterogeneous distributed environments that require intensive data exchange among context bubbles but run on various platforms and speak different vernaculars. Within the ARUM context, such an environment is realised by designing a SOA-inspired platform that is primarily designed to support ramp-up systems for manufacturing, where there are conflicts between the need for control and rigour and the reality of rapid changes. To address such challenges and industrial requirements the architecture of ARUM platform integrates the key features of service-oriented-architecture, holo-

nic multi-agent systems and legacy systems and links them via an enterprise service bus (ESB) [1], providing communication, monitoring, interoperability and aggregation of information across existing legacy systems at all production levels to support real-time automatic negotiation, planning, scheduling and optimization within and across factories. The envisaged technologies of distributed multi agent system within a holonic architecture is expected to help in integrating legacy systems, information aggregation from high level systems (MES, ERP, etc.) to factory floor automation (e.g. metal cut and assembly systems). Based on such requirements reflected in the system architecture, the main functionalities of ARUM platform include scheduling, planning, production management and manufacturing process supported by information delivered from a variety of sources e.g. legacy systems, sensors, and users.

The use of multi-agent system and service-oriented architectures in manufacturing is not new, and other approaches have already been promoted, mainly focusing the lower-level control and interaction [10, 0]. Despite this, the ARUM project differentiates from those, by promoting the integration of different layers of the ISA-95, namely the planning and scheduling, aggregated by the use of a common communication infrastructure and a well-define data access process flow.

The rest of the paper is organized as follows: Section 2 presents the ARUM architecture, briefly describing its components and tools, while Section 3 describes the technical background that supports that architecture. Section 4 describes, by means of two practical examples, the usage of the ARUM architecture, particularly the data retrieval and the strategic planning actions. At last, Section 5 rounds up the paper with the conclusions.

2 ARUM Platform Architecture

Based on the user and domain specific requirements as well as the underlying SOA principles, the architecture of the ARUM platform is closely coupled with the goal of enabling prediction in the pre-planning phase and real-time control in the production phase in highly dynamic ramp-up production domains. Figure 1 depicts the system architecture and its key components and services that make up the ARUM platform, as well as the interrelations between them, making it easier to understand the functionality offered and realise the potential benefits of the ARUM platform.

As shown in Figure 1, the SOA inspired **Enterprise Service Bus** constitutes the core of ARUM intelligent ESB (i-ESB) platform and provides a common communication infrastructure for the interoperability of different tools and services, such as the Strategic Planner, Ontology Service, FND&SD service and others described below.

The Ontology Service and the MIDAS/Azimov tool are considered as the **knowledge base** of the system. The **Ontology Service** is the major data provider to the ARUM tools as it holds the schema of the ARUM Ontologies (Core, Scene, Events and Policy Model) and manages production-related data expressed based on the appropriate schema in a triple store, a special data structure for semantic data.

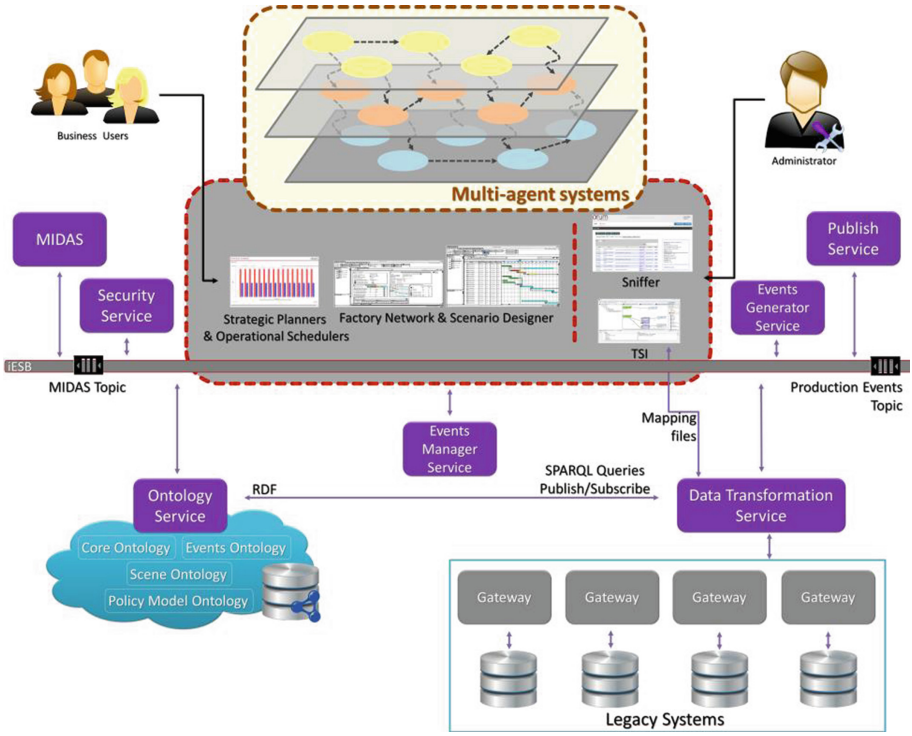

Fig. 1. ARUM system architecture

The Legacy Systems, the ARUM tools and services, as well as the Production Events Topic are the three data sources from which data are acquired by the Ontology Service. In the first case, data in various formats coming from the Legacy Systems on demand by the aforementioned tools are converted into RDF by the Data Transformation Service before sending them to the Ontology Service. In the second case, semantic data resulting from scheduling and planning processes operated by the related tools, such as the current state of the factory used by the Operational Scheduler for what-if games, are sent to the Ontology Service for the purpose of storage and/or reasoning. Finally, in the third case, each iESB service publishes any production-related event in the Production Events Topic by exploiting the schema of the ARUM Events Ontology. The **MIDAS/Azimov** [8] tool receives the production-related events via the **MIDAS Topic** supported by a publish/subscribe messaging model and analyses them by taking also into account their respective solutions. In this respect, all information is processed and in order to facilitate the troubleshooting procedure the estimated troubleshoot time is provided for given production events, typically non-conformities in the production/manufacturing processes.

Day-to-day data of current manufacturing practices contained in existing systems known as **Legacy Systems** are integrated in the platform via gateways of the **Data Transformation Service,** which transforms data from heterogeneous sources (e.g. legacy systems) into the RDF format according to the ARUM ontology. The mapping

between the schema of legacy systems data and the ARUM ontology, based on which the Data Transformation Service performs the transformation of the data, is created by the **TIE Semantic Integrator** (TSI).

An innovative feature of the ARUM architecture is the support for multiple iESB services that serve generic purposes providing support for the whole infrastructure. To be more specific, the **Security Service** is mainly responsible for authenticating users as well as making access control decisions based on policies specified by the Policy Model Ontology when. This service plays a crucial role to the ARUM infrastructure by acting as a mediator in the communication for data exchange between the iESB services, thus ensures trust. The **Publish Service** used by other services to distribute messages in a publish-subscribe manner posts any received message to a corresponding topic to which other services are subscribed. The **Event Generator Service** used by the Scenario Designer tool for obtaining generated hypothetical production-related events on request is useful in what-if game simulations and in this way it supports planning in the manufacturing domain. The **Production Events Topic,** as mentioned above, receives production-related event messages by an ARUM console, legacy systems, or any other service such as FND&SD.

The main tasks of the ARUM platform regarding scheduling and planning are carried out by **Multi-Agent Systems** [6] **as a set of intelligent services (MAS realm)**. These services mainly carryout the scheduling and planning activities, which effectively are the main features of the ARUM platform. More specifically, the **Operational scheduler** as a tool computes one or more valid schedules as a set-up of start and end times of (manufacturing) jobs and allocation of resources based on input data about jobs to be done, their precedencies and required resources. The **Strategic Planner** [3] assists the professionals taking managerial decisions, particularly at the tactical level by enabling them to identify and analyse which and where resources must be allocated in order to cope with production demand as well as at the strategic level by helping them decide if, e.g., building an extra production line is required.

The presentation tier of the ARUM platform, based on open source technologies and products (such as Apache Rave and Wookie) that can support widget based environments or LifeRay for portlet-based dashboards, helps in creating an iterative user-centred experience. Moreover, the **Factory Network and Scenario Designer (FND&SD) Service** acts as the gateway between the ESB and the user interface. Its GUI provides users with functionalities such recording and processing production events. Finally, the **ESB Sniffer** GUI [7] provides a list of messages going through the ESB together with their details, showing a visualization of the message flow, being particularly useful for debugging problems in inter-service communications.

3 Architecture Support for Technical Solution

The support from the above described SOA-inspired architecture enables ARUM platform to bring together a new generation of innovative *Multi-Agent Systems (MAS) that can build plans and schedules in real time* based on the information collected from different resources including legacy systems (e.g., MES, ERP, etc.), data gener-

ated by workers, sensors, MAS themselves, etc. and *enterprise integration platforms that follow different paradigms* in order to guarantee fast delivery of business value, maximal coverage of systems accepted by the platform, and a security - the high-level overview of ARUM platform is shown in Figure 2.

An extra challenge of ARUM platform lay in the fact that MASs need a massive flow of information into them up-front and continuously during their execution to make their work effective. This is a well-known so-called "knowledge bottleneck" problem but in case of ARUM velocity and volume of data are also playing an important role since a schedule for workers that does not factor in the latest figures over the articles in the local warehouse or the outcomes of the recent quality checks would have no value. And to make it even more challenging providers of business tools in ARUM requested platform providers to support ontologies and triple stores for the convenience of their communications among each other and with other data sources. This solution also helps them to focus on their functionality and delegate all the heavy lifting work to the platform.

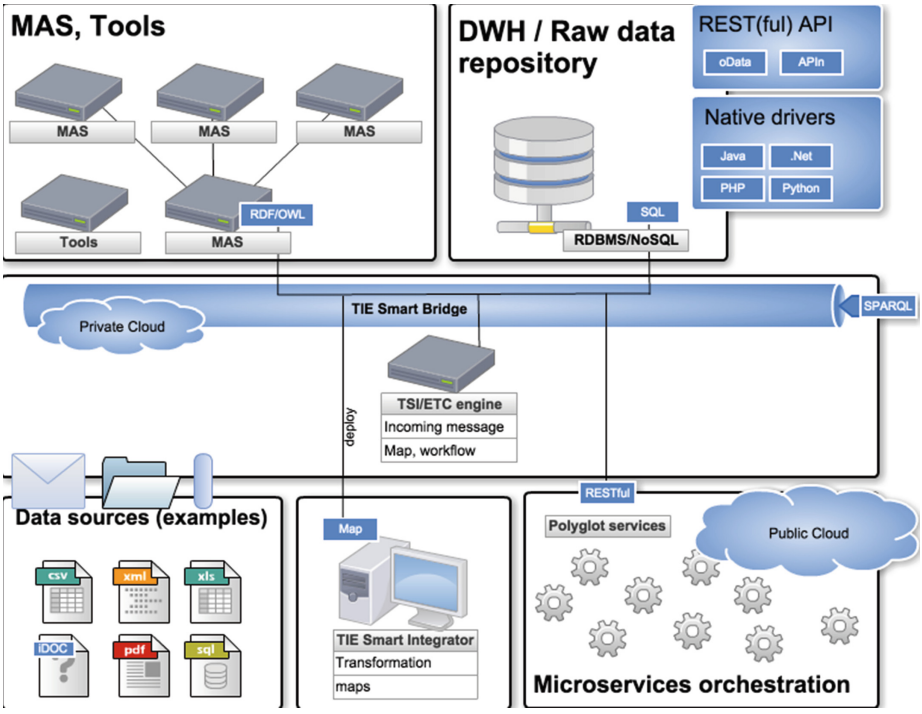

Fig. 2. High-level overview of ARUM platform

The architecture also support ARUM platform to take advantage of the realm of Internet of Things and Knowledge-based systems such as Multi-agent systems (MAS) and thus consists of many independent services and data sources that reside on various platforms spanning from legacy monoliths to notebooks, and tablet PCs. Coopera-

tion via interaction of all those systems (hardware and software) is the intrinsic characteristic of the ARUM platform. Integration with legacy and off-the-shelf monolith systems is important because tones of valuable data are already there and those systems are tightly integrated into the enterprise business processes and they will not be replaced any soon. Legacy nature of those systems (such as CAD, CRM, HR, etc.) also means that those systems most of the time does not have common interaction mechanisms, Web Service interface and cannot communicate via HTTP protocol. As mentioned above, the the ARUM Intelligent ESB (iESB) platform employs a combination of two message-service busses, namely TIE Smart Bridge (TSB) and JBoss ESB. This combination was suits the requirements since it allows interoperability and interplay between different programming paradigms (e.g. .Net and Java) that cover most of the traditional enterprise integration spectrum.

This also allows integration of different proprietary and open source products. In addition to the combination of two service buses, any other ESB can be connected to the platform that follows ARUM approach of a shared stack of message queues and common service registry. Because queues are hosted in the cloud environment it is possible to dynamically manage them, for instance, in order to react to the spiky loads. A service registry supports a dynamic discovery, which adds agility to the whole infrastructure by allowing on-the-fly discovery and service binding. Of course, all other ARUM added-value services that constitute the complete iESB are also deployed in the cloud and can benefit from using the same dynamic discovery and elasticity along with a polyglot persistency and service execution environments. This become especially valuable when ARUM had faced a need of integrating new wave of service and data providers that reside on the hand-held devices of the shop floor personnel such as tablets or notebooks used by workers and managers next to the working stations.

An example scenario for such integration is the interplay of TIE Smart Bridge (TSB), a lightweight message and service bus that strongly supports SOA principles, with TIE Smart Integrator (TSI), a semantic mapping and transformation tool. TSB has a history of successful business brokerage for different sizes of commercial projects and a rich set of off-the-shelf functionality for integration with a wide spectrum of legacy systems as well as with services and data providers that are hosted in the cloud. TSB relies on TSI for doing explicit declarative maps that describe how elements of a source schema correspond to elements of a destination schema. Transformation engines use those maps in the data exchange and integration scenarios. Both of these tools can be exposed and used via REST APIs, and altogether this solution paves the road between two integration paradigms, SOA and REST.

A typical scenario starts when a factory-floor worker sends a message via a specialized mobile application to the platform about finishing his or her job. This message is accepted by the data warehouse using a REST API, and the appropriate data is added or updated. Other devices or software services that are connected through RESTful APIs or use native drivers, such as ODBC or JDBC (see Figure 1), can get access to this new state right away. But that is not all, since ontologies have to be supported to enable meaningful information exchange between different components and systems. Consequently, a service responsible for managing knowledge within the

platform, namely the Ontology Service, is subscribed to a publish-subscribe endpoint in TSB that can manage different queries in SPARQL format – a W3C standard for querying Web ontologies. In this respect, the Ontology service can manage internal logic and requests from its main clients, namely planners, schedulers, and other knowledge-based tools.

Although in theory any service can subscribe to this pub/sub point, the only one practical subscriber at the moment is the Ontology Service. The Ontology Service focuses on providing a clean API and language understood by knowledge-based systems; it intensively manipulates knowledge in a triple store, searches for implicit relationships using different inference mechanisms, and it is ready to reply to complicated analytical queries.

Furthermore, in order to satisfy the needs of knowledge-based systems, TSI was enriched to deal with the transformation of data into the Web ontology format OWL. Thus when a SPARQL request arrives:

1. It is rewritten into one or more SQL queries.
2. Data is retrieved from the target systems.
3. It is transformed using existing maps into the destination format.

All three steps are implemented via micro-services, and they can be executed in parallel when possible. Implementation of a scale-out strategy here also helps in increasing the throughput of the system and decreases its latency. A similar situation exists between the platform's data warehouse and legacy systems. TSB is used to subscribe or to poll legacy resources through its gateways. When data is emitted by any of the data providers, TSB starts an appropriate workflow that contains necessary transformations, and data is finally pushed into the data warehouse.

The above scenario describes a normal operational routine of our solution, when data comes in small chunks from different directions. There is another mode used for the initial setup. In this mode, TSB collects data from the connected legacy systems, transforms it using TSI and stores it in the data warehouse, and then performs a massive batch transformation into ontological format followed by the upload of that data into the Ontology Service. This is necessary so that a new system does not suffer from a "cold start" — a term used in knowledge-based systems to describe a situation in which they cannot make any meaningful suggestions due to a lack of available information.

Together TSB and TSI can be provided within the cloud as software-as-a-service (SaaS), and they can help in integrating business services dispersed all over the Internet running on their own distributed virtual machines (VMs) and virtual environments (VEs; e.g., Docker) with a diversity of operating systems. Thus, the proposed solution speeds up and facilitates delivery of the new business value by quickly setting up a foundation for communications in heterogeneous distributed environments that require intensive data exchange among context bubbles but run on various platforms and speak different vernaculars

4 Usage Scenarios

We have carried out the validation of the proposed architecture and the resulting platform in the context of ensuring that architecture supports the perceived functionalities and the platform can handle the system level operations with the help of the following data retrieval scenarios.

Example of the interaction of the Factory Network & Scenario Designer with the Ontology ServiceIn this scenario (as shown in Figure 3), a user of the ARUM platform, such as a Station Manager, wants to design a new scene, i.e., a new snapshot of the current state of the station and associate it with a particular product, which is processed at the aforementioned station. This operation can be performed through the U**ser Interface (UI) of the Factory Network & Scenario Designer (FNDSD)**.

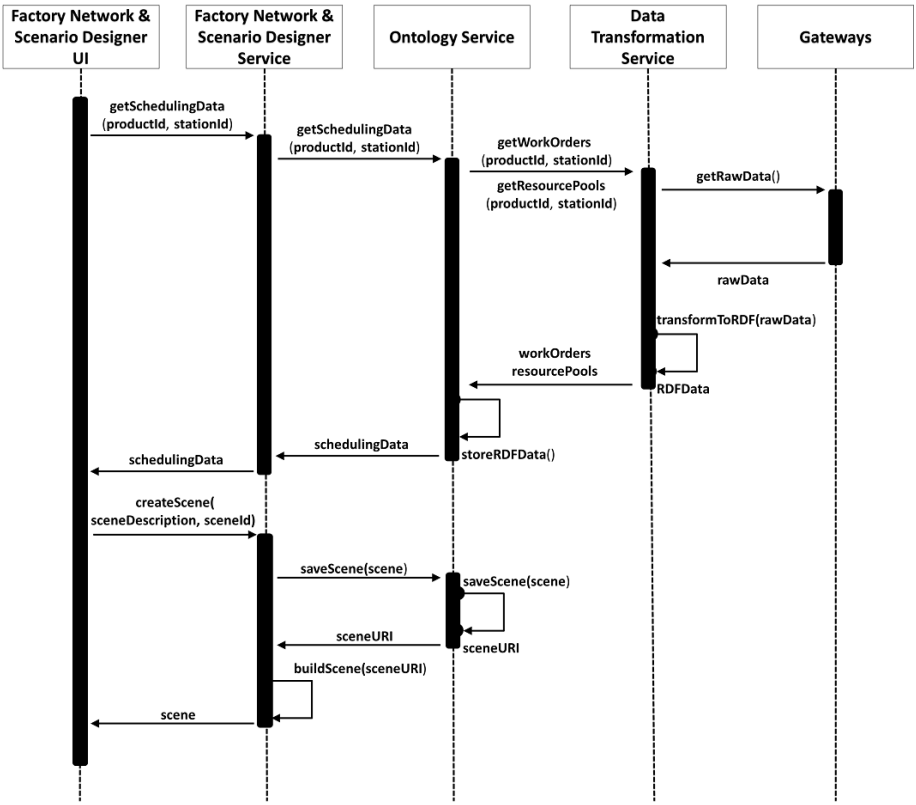

Fig. 3. Sequence diagram for the legacy systems data retrieval scenario

Before the creation of the scene is possible, certain pieces of data, which will be later used as part of the scene during the scheduling process, need to be acquired. Once such a request is sent by the FNDSD UI to the **FNDSD service**, the latter sends the appropriate request to the Ontology Service. Since, initially, the legacy data are

kept in the legacy systems, the FNDSD service requests the Ontology Service to fetch all data that are needed for the scheduling of the operations concerning the processing of the aforementioned product to the station, where it is allocated.

In this respect, the **Ontology Service** decomposes the query into sub-queries that are then sent to the Data transformation service. The reason for decomposing the initial query is that various pieces of data are spread over different legacy systems. Basically, the requested data contains the set of operations to be performed, their precedencies, durations, required human resources with particular skills, required parts, availability of resources, etc. This communication does not go over the ESB, but rather via a Web service based API of the Data Transformation service. In particular, the requests are reflected by means of SPARQL queries.

Then, the **Data Transformation Service** issues several requests to appropriate Gateways, each of which is designed to retrieve data from particular legacy MES or ERP system. The retrieved data are returned back to the Data Transformation Service in its native format. The Data Transformation Service then converts the acquired data into the semantic format (RDF) using semantic mappings generated by TIE Semantic Integrator and the ARUM Core and Scene ontologies as target schemas. The resulting pieces of RDF data are returned back to the Ontology Service.

When the Ontology service receives the RDF data, it aggregates it in a dedicated triple store and then returns it to the FNDSD service. Local storage is intended for easy and faster data access in case such data is requested again.

Having received the scheduling data, the FND&SD service returns it to the FNDSD UI, which is now able to issue a request to create a new scene. This request is sent back to the FNDSD service, which in turn asks the Ontology Service to save it in the triple store. The result of the action performed by the Ontology Service is the generation of a new unique URI for the scene, which is sent to the FNDSD service. The latter builds a scene and sends it back to the FNDSD UI. The scene will be then used as a holder of input data for the Scheduler.

4.1 Strategic Planning Operation

The higher hierarchical levels of any manufacturing company must take strategic decisions that culminates, among others, in the construction of a new production site or in a shop-floor re-organization, e.g., by means of the use of additional degrees of freedom (DoF), such as the introduction of extra working hours or of a new production line.

In such way, the Strategic Planner (SP) in the ARUM platform combines the agility and flexibility provided by the software agent technology with the optimization levels provided by classical mathematical models [5] – as shown in Figure 4. The SP tool is a bundle composed of a User Interface (UI) and a Multi-Agent System (MAS) that encapsulates the mathematical solver, as depicted in Figure 4. Both components interact within the ARUM platform using a standardized message mechanism that combines the standard service communication process with the interaction protocols facilitates offered by the agent world, namely the FIPA (Foundation for Intelligent Physical Agents) interaction protocols.

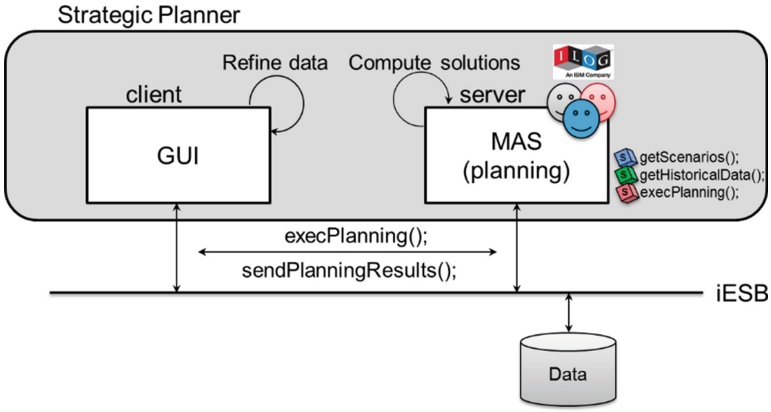

Fig. 4. User interface and multi-agent system planning interaction

The planning process, each time the user wants to use the tool, in reaction to an external event or to perform exploratory studies, starts by the request of the available data, i.e. the system current status. After this, the user designs the conditions to be tested, e.g., a product demand increase or costs variation, and requests the computation of a solution. At this stage, the agents within the MAS cooperate in order to provide the requested plan with the details of associated tasks, activities and resources - leading to the solution (or a set of best solutions) [9].

5 Concluding Remarks

ARUM is aimed at the development of a flexible and adaptive ICT solution for production management and control of highly complex, small lot productions such as in aircraft and shipbuilding industries. Small lot manufacturers have to deal with specific challenges such as the high investments in the product design and ramp-up due to the complexity of the final product and very small production batches.

This paper provides an overview of the ARUM platform that uses SOA-inspired design principles to narrow the gaps that current practices cannot do in small-lot production and ramp-up of highly customized and complex products. The key objectives of ARUM platform include ensuring purposeful and unobstructed flow of information between decision makers, intelligent planning, scheduling, optimization, control tools and legacy systems.

To support the stakeholders in small lot manufacturing and complexity of product design and ramp-up stages, the architecture of ARUM platform presented in this paper promotes the view of services as Intelligent Services that are defined as independent pieces of software and are expected to provide a particular result either produced by the intelligent services themselves or by requesting support from other intelligent services. Seamless communication between intelligent services is supported by adopting the SOA-inspired Enterprise Service Bus solution. The use of a combination of ESBs in ARUM platform also facilitates other operational features such of knowledge acquisition and sharing by exploiting different ontologies, integration of legacy sys-

tems, monitoring of dynamic interactions by reviewing the flow of high volume of message exchanges, service lifecycle management and platform distribution.

This paper does not cover the details or research behind the scheduling, optimisation and planning of production services. Functionality details and software specification of these services and other ARUM aspects can be found by exploring different pieces of work in the references section.

In our future work, we are performing empirical and user-based assessment of the platform and associated services in order to determine their effectiveness in delivering the perceived benefits.

References

1. Marín, C.A., Mönch, L., Liu, L., Mehandjiev, N., Lioudakis, G.V., Kazanskaia, D., Chepegin, V.: Application of intelligent service bus in a ramp-up production context. In: CAiSE 2013, Valencia, Spain, June 17-21, 2013
2. Inden, U., Mehandjiev, N., Mönch, L., Vrba, P.: Towards an ontology for small series production. In: Mařík, V., Lastra, J.L.M., Skobelev, P. (eds.) HoloMAS 2013. LNCS, vol. 8062, pp. 128–139. Springer, Heidelberg (2013)
3. Leitão, P., Barbosa, J., Vrba, P., Skobelev, P., Tsarev, A., Kazanskaia, D.: Multi-agent system approach for the strategic planning in ramp-up production of small lots. In: SMC 2013, Manchester, UK, October 13-16, 2013
4. Marin, C., Moench, L., Leitao, P., Vrba, P., Kazanskaia, D., Chepegin, V., Liu, L., Mehandjiev, N.: A conceptual architecture based on intelligent services for manufacturing support systems. In: SMC 2013, Manchester, UK, October 13-16, 2013
5. Biele, A., Mönch, L.: Using a math-heuristic to optimize mixed model assembly lines in low-volume manufacturing. Informs Annual Meeting, Minneapolis, Minnesota, USA, October 2013
6. Leitão, P., Barbosa, J.: Adaptive scheduling based on self-organized holonic swarm of schedulers In: ISIE 2014, Instanbul, Turkey, June 1-4, 2014
7. Vrba, P., Myslik, M., Klima, M.: JBoss ESB sniffer: message flow visualization for enterprise service bus. In: ISIE 2014, Instanbul, Turkey, June 1-4, 2014
8. Stellingwerff, L., Pazienza, G.E.: An agent-based architecture to model and manipulate context knowledge. In: Demazeau, Y., Zambonelli, F., Corchado, J.M., Bajo, J. (eds.) PAAMS 2014. LNCS, vol. 8473, pp. 256–267. Springer, Heidelberg (2014)
9. Ferreira, A., Pereira, A., Rodrigues, N., Barbosa J., Leitão, P.: Integration of an agent-based strategic planner in an enterprise service bus ecosystem. In: 13th IEEE International Conference on Industrial Informatics (INDIN 2015), Cambridge, UK, July 22-24, 2015
10. Rocha, A., di Orio, G., Barata, J., Antzoulatos, N., Castro, E., Scrimieri, D., Ratchev, S., Ribeiro, L.: An agent based framework to support plug and produce. In: 2014 12th IEEE International Conference on Industrial Informatics (INDIN), vol. 504, no. 510, pp. 27–30, July 2014
11. Karnouskos, S., Colombo, A.W., Bangemann, T., Manninen, K., Camp, R., Tilly, M., Sikora, M., Jammes, F., Delsing, J., Eliasson, J., Nappey, P., Hu, J., Graf, M.: The IMC-AESOP Architecture for Cloud-Based Industrial Cyber-Physical Systems. In: Colombo, A.W., Bangemann, T., Karnouskos, S., Delsing, J., Stluka, P., Harrison, R., Jammes, F., Lastra, J.L. (eds.) Industrial Cloud-Based Cyber-Physical Systems, pp. 49–88. Springer International Publishing, Cham (2014)