Inf Syst E-Bus Manage (2018) 16:683–720
https://doi.org/10.1007/s10257-018-0371-5

ORIGINAL ARTICLE

# A model-driven software engineering workflow and tool architecture for servitised manufacturing

**Emmanouil Ntanos[1] · Gerasimos Dimitriou[2] · Vassilis Bekiaris[2] · Charalampos Vassiliou[2] · Kostas Kalaboukas[2] · Dimitris Askounis[1]**

Received: 25 July 2017 / Revised: 24 January 2018 / Accepted: 2 March 2018 /
Published online: 6 March 2018
© Springer-Verlag GmbH Germany, part of Springer Nature 2018

**Abstract** Modern manufacturing businesses increasingly engage in servitisation, by offering advanced services along with physical products, and creating "product–service systems". Information Technology infrastructures, and especially software, are a critical part of modern service provision. However, software development in this context has not been investigated and there are no development methods or tools specifically adapted to the task of creating software for servitised businesses in general, or manufacturing in particular. In this paper, we define the requirements for software engineering in servitised manufacturing. Based on these, we describe a model-driven software engineering workflow for servitised manufacturing, supporting both structural and behavioural modelling of the service system. Furthermore, we elaborate on the architecture of an appropriate model-driven Integrated Development Environment (IDE). The proposed workflow and a prototype implementation of the IDE were evaluated in a set of industrial pilots, demonstrating improved communication and collaboration between participants in the software engineering process.

**Keywords** Product-service system · Software engineering · Model driven · Tool integration · Integrated Development Environment · Manufacturing

✉ Emmanouil Ntanos
entanos@epu.ntua.gr

[1] School of Electrical and Computer Engineering, National Technical University of Athens, 9 Iroon Polytechniou Str, 15780 Athens, Greece

[2] SingularLogic Software, Al. Panagouli & Sinisioglou Strs, 14234 Athens, Greece



المنارة للاستشارات

## 1 Introduction

Product-oriented firms are undergoing a process of "servitisation", where physical goods and services are becoming increasingly entwined, as equally important parts of a combined offering (Vandermerwe and Rada 1988), a strategy often driven by competitive pressure (Neely 2008). Such "Product–Service systems" (PSS) may include, for example, smart home devices, fuel economy assistance for automobiles (Saarijärvi et al. 2014), remote Prognostics and Health Management (PHM) for industrial equipment (Guillén et al. 2016; Vogl et al. 2016) and others. Information Technologies (IT) play a crucial infrastructural role in service provision and PSSs. They enable the deployment of advanced services through capabilities such as business automation, networked communication, data analytics, and personalisation (Boehm and Thomas 2013; Lim and Kim 2015; Abramovici and Filos 2011).

For example, a firm manufactures and sells domestic heat-pumps to resellers who install them in homes and provide technical support and maintenance. The resellers are undertaking ad-hoc repairs each time the end-customer notices a malfunction and requests a service appointment. The manufacturing firm is considering "servitising" its equipment offering by offering a subscription-based equipment monitoring and maintenance service. In this scenario, the heat-pump's sensor module will monitor its operation and will transmit operating parameters and fault codes to the manufacturer's servers via the web. Once a fault (or potential fault) is detected by the server, the local reseller will be alerted via email with an automated fault report, and will contact the end-customer to arrange a service appointment. When on-site, the technician will retrieve diagnostics directly from the server on a tablet, perform maintenance and submit a maintenance report or an order for spare parts back to the central system.

Realising this service offering requires the operation and coordination of a complex Information Technology system. The heat-pump itself must be connected to the web and transmit information to a server. A decision-support system will apply rules and heuristics to flag possible malfunctions. If faults are found, the alert reports must be generated and mailed to the reseller network. Once at the site, the technician will need to access technical data from the device, and submit a report online. Billing and subscription management requires a Customer Relation Management (CRM) system.

While software is a vital part of the IT infrastructure of servitisation, its development as part of the infrastructure of a larger Product–Service System is largely unexplored. There is no theoretical or empirical work addressing the conditions and problems encountered by enterprises and practitioners in software engineering for service systems. This contributes in the creation of a substantial research gap in the areas of methods and tools for developing applications within the framework of a "Software-Product-Service" system (Mikusz 2014) or for the design of "informatics-based services" in manufacturing (Lim et al. 2015).

As a response to this gap, research was conducted in the framework of the "MSEE: Manufacturing SErvice Ecosystem" project, co-financed by the



المنارة للاستشارات

European Commission's 7th Framework Programme. Using an iterative approach, the elements of a suitable workflow and tool architecture and a prototype implementation of an Integrated Development Environment (IDE) were refined, and then evaluated in the project's industrial pilots.

In this paper we propose a model-driven software workflow and describe the architecture of the necessary tools for the development of software for PSS. Specifically, we examine the task of implementing software in the conditions imposed by the business environment of servitisation. Based on this review, we propose a suitable workflow based on the Model-Driven Service Engineering Architecture (MDSEA), a generic model-driven framework for service system design. Furthermore, we describe the architecture of a Development Platform capable of supporting the proposed workflow. Finally, we report on the evaluation of the workflow and a prototype development environment in a set of industrial pilots, by developers working in and with manufacturing enterprises.

The rest of the paper is organised as follows: Sect. 2 presents current work in the fields related to this research, and Sect. 3 describes the research methodology used. Section 4 presents the proposed software development workflow, while Sect. 5 describes the reference architecture of a suitable development platform. The evaluation process of the methodology and the proposed architecture are described in Sect. 6, while Sect. 7 concludes this paper and suggests avenues for further research.

## 2 Related work

The motivating problem involves addressing a contemporary and evolving business need with methodological and technological tools from Computer Science and Information Technology. Outlining this need, and determining the tools to address it, require understanding the current business and technological landscape.

### 2.1 Service systems

A service system can be defined as a "dynamic configuration of resources for the co-creation of value", containing people, organisations, shared information and technology, and connected internally, as well as externally, with other service systems (Maglio et al. 2009; Vargo et al. 2008). The collaboration of multiple business partners is necessary for the delivery of advanced and complex services in servitisation (Johnson and Mena 2008; Gao et al. 2011). The "Product-Service Systems" (PSS) offered by modern servitised manufacturing involve a network of manufacturers, service providers, business and technical personnel, business partners and customers (Meier et al. 2011; Bikfalvi et al. 2013; Reim et al. 2015). Creating a new service system requires the involvement of organisations and personnel from different backgrounds and specialisations, both technical and non-technical (Brax and Visintin 2016).

The creation of a service system involves the parallel co-development of all its aspects, such as organisational structures, processes, human resources, IT, physical

المنارة للاستشارات

locations, or products. Its design should include the roles of people, technology, physical facilities, equipment and processes (Goldstein et al. 2002). For example, the provision of technical support services for new equipment requires defining the types of services to be provided, the support processes themselves, the technicians and other personnel implementing them, how they are organised in teams (e.g. by location or specialty), where they will be based, the necessary infrastructures (warehouses, offices) and resources (spare parts, consumables). In another context, implementing an application store for mobile devices requires the identification of business processes, personnel (e.g. administrators, quality control personnel, or software developers), physical infrastructures (e.g. offices) and IT infrastructures capable of serving multiple mobile device users. The service system will not operate correctly (or at all), if any of these elements is missing or is badly implemented. In the case of Product-Service Systems, service and product functions must be integrated from the early stages of design, and designers should consider the entities involved, the system lifecycle, and the corresponding actor networks (Cavalieri and Pezzota 2012).

Technology-enhanced service provision also takes place in various "contexts": person-to-person services, self-service, multi-channel services, multiple devices and platforms, back-stage services, and location/context aware services (Glushko 2010; Lim et al. 2015). This implies a variety of technologies and platforms (e.g. embedded systems, mobile devices, or point-of-sales terminals), and applications (such as web sites, databases, decision support systems and financial transaction systems). In the case of software for service systems, development cycles are driven by the evolution of the corresponding service offerings due to competitive pressures. Service software is never "finished" as long as the service system it supports keeps evolving (Chae 2014). In this environment, the ability for prototyping and reducing "time-to-market" is essential (Van Riel and Lievens 2004; Mietinnen et al. 2012). The need for collaboration has been observed in several aspects of software development for manufacturing and service systems, for example in developing the security aspects of supply chain management software (Bartol 2014), software-intensive product development (Pernstål et al. 2015), and the automotive industry (Lim et al. 2015).

Based on the above, it is possible to draw a set of conclusions regarding software engineering for service provision and product-service systems. There is a need to coordinate several stakeholders in the design and implementation of the service system. These stakeholders include the software engineers who will be tasked to design and implement the software supporting the provision of services, and they will receive input from several sources (business analysts, service network partners, and others). Additionally, they may need to leverage a large variety of technologies depending on the way the end services are provided, thus requiring methodologies to translate these diverse requirements into applications.

### 2.2 Service-oriented systems and business process management

A common thread of modern enterprise IT for business service provision is the application of Service-Oriented Architecture (SOA) principles, i.e. the composition of software components (services) to create complex applications (Rosen et al.



المنارة للاستشارات

2012; Huhns and Singh 2005; Papazoglou and Georgakopoulos 2003). Additionally, most current practical SOA implementations are based on web services standards and technologies (Rosen et al. 2012) such as REST, SOAP, XML, WSDL and others. SOA-style systems, and especially service compositions, are closely associated with Business Process Management (BPM) technologies, which aim to model business processes as sequences of tasks (including the invocation of SOA-style software services) using abstractions such as the Business Process Model and Notation (BPMN) (White 2008) and others. Because of this emphasis on abstraction, model-driven engineering approaches are suitable for the development of SOA and BPM systems, with considerable research being done in this field (Ameller et al. 2015).

SOA and BPM aim to improve the alignment of business processes with software by matching the elements (stakeholders, processes, entities, relations and others) of the business domain. Additionally, the modular and "loosely-coupled" nature of service-oriented and BPM systems means that they can be easily reconfigured to produce new complex functionality, and quickly respond to customer needs and competition. Finally, SOA and BPM are mature and widely accepted Information Systems paradigms. Therefore, when considering the IT environment of service provision, it is reasonable to anticipate the use of Service-Oriented Architectures and Business Process Management technologies.

### 2.3 Software engineering tool integration

Software engineering is not the main value driver of a physical product-oriented firm. In-house software development capabilities may be limited (Wallin et al. 2015) or focused on fields other than service or enterprise applications. IT personnel may also have limited access to appropriate tools and know-how. Often, software development is outsourced to another, more specialised, organisation, requiring effective communication and collaboration channels. As such, product-oriented firms may need to acquire the proper tools for software engineering or for participating in such a process (e.g. to collaborate with software contractors). The requirements and software engineering workflow needs to be easy to set up, with smoothly interoperating stages and tools, while providing flexibility for all required end-products (for example, applications, services, service compositions, and databases). This indicates the need for the application of "software engineering tool integration" principles.

Tool integration is the practice of integrating software development tools, in an effort to streamline the development process. This integration may include "vertical tools" which handle specific steps of the development workflow (such as modellers, code editors, and compilers) and "horizontal tools" which provide support for the overall process (such as documentation, and project/workspace management) (Wasserman 1990; Thomas and Nejmeh 1992). Integrated Development Environments (IDEs) represent the highest level of tool integration in software engineering. In this context, it is reasonable to assume that software development in service systems and servitised manufacturing would be well served by the use of integrated tools, and especially IDEs. Integrated tools are easy to set-up and can be adapted to particular

المنارة للاستشارات

domains and the technologies used for service provision. IDEs can also contain collaboration tools such as common repositories, dashboards, and messaging.

### 2.4 Product service system design methodologies and IT

PSS design techniques generally do not provide explicit support for the development of service software. Becker et al. (2010) concluded that modelling languages and methods that do not originate from an Information Technology background do not capture the information systems and IT requirements of service systems. Yet, IT-derived approaches do not capture the details of customer interaction, value creation or of the product itself. Berkovich et al. (2009) acknowledged the need for the parallel development of the product, service and the required software but concluded that there are no mature methodologies, and further research is necessary. In 2011, the same authors presented a framework for requirements engineering for PSS development, and noted that future research could combine elements from software, product, service and PSS fields for a more comprehensive approach (Berkovich et al. 2011). In 2012, Vasantha et al. analysed a number of relatively mature PSS design techniques in order to detect methodological gaps but none provided specific guidance for the development of software and information systems (Vasantha et al. 2012). The state-of-the-art review performed by Qu et al. (2016) listed 36 PSS design and development methodologies in cited research. However only three of those appear to make any mention of Information and Communication Technology aspects of the PSS.

Even if they explicitly acknowledge the information systems involved, PSS design methodologies focus on a high-level conceptual view of the service system design or on business aspects. IT requirements, functions or development activities are simply mentioned but are not analysed in technical or methodological terms. This includes several approaches: Morelli (2002) outlines the functional modelling of a PSS, which only mentions the activities which are performed by IT infrastructures. Aurich et al. (2006) presented a systematic PSS design methodology in which the design of the business functions may lead to requirements for data interchange, but provided no further guidance on how these will be expressed and translated into working systems. Maussang et al. (2007) presented a PSS design methodology where high-level requirements are translated into "Functional Block Diagrams" of service activities, including IT-based activities, but without discussing their transformation into concrete requirements. Lindström et al. (2012) describe a process for the development of "Functional Products" which also include hardware, software and a service support system, but only focus on the project management decision stages of the development. Zhao and Cai (2013) provided a draft and a metamodel for model-driven PSS building methodology based on MDA concepts, including an IT aspect, but without elaborating it further. A project management-level methodology for Industrial PSS ($IPS^2$) by Nguyen et al. (2014), simply included software as one of many steps to build a PSS. Berkovich et al. (2014) described a PSS Requirement Data Model, in which software engineering requirements are mentioned, but there is not further information on how they should be expressed, processed and



المنارة للاستشارات

transformed into functional applications. Stokic and Correia (2015) propose the conceptual architecture of an engineering environment for extending manufacturing products via web-based services. Its elaboration and development is identified as one of the goals of a wider research project for service engineering research ("DIVERSITY"—FP7 Factories of the Future, Grant 636692). A follow-up publication describing the engineering environment designed by the project includes no mention of software development as it focuses on high-level product-lifecycle management (Pezzotta et al. 2016). Correia et al. (2017) presented an ontology for communication of stakeholders and tools in the development of a PSS, which contains several concepts, including "software" but without further analysis of the concept. The method proposed by Metzger et al. (2017) included software development, but focused on how this process will be carried out from a project management perspective.

There is an apparent methodological gap in integrating software development in the product-service system engineering process. In the current literature there are no concrete methods for integrating software development in the PSS development process, even if the software component is recognized as an important aspect. A notable exception is the Model-Driven Service Engineering Architecture framework (MDSEA) described in the following section.

### 2.5 Model driven engineering for service systems: the MDSEA framework

In software development, Model Driven Engineering (MDE) aims to reduce the distance between the application domain and the software under development by using models that describe a system at various levels of abstraction and from different viewpoints. MDE also facilitates communication between domain experts (e.g. engineers and business analysts) and software engineers building software for the company's core products, services and activities (Hutchinson et al. 2011).

MDE has proven useful to organisations developing software in business domains outside software engineering. The application of "meta-models" i.e. models about the entities, relationships and interactions in an application domain (as opposed to a specific application) simplifies and standardises modelling approaches. The HL7 v.3 set of healthcare informatics standards (Health Level Seven International 2017) is a relevant example used as a foundation for research on model-driven methodologies and tools in the healthcare domain. For instance, research in this area resulted in a methodology to link the HL7 metamodel to standardised Unified Modelling Language (UML) class diagrams (Martínez-Garciá et al. 2015), a development framework for interoperable healthcare applications (Lopez and Blobel 2009), the application of Model-Driven Architecture (MDA) modelling and transformation methods for software design based on the HL7 metamodel, and the automated generation of code for web services and service orchestrations using HL7 as part of the application domain layer (Anzböck and Dustdar 2005).

Model-driven approaches can be used to model and create software for service provision software. Service systems are a special case of enterprises, and Enterprise Modelling requires the distinction between structural aspects and



المنارة للاستشارات

behavioural aspects of the enterprise system (Jonkers et al. 2005). Structural models of a domain contain entities, relations and concepts (e.g. entity classes). Behavioural models of a domain contain actions, events and changes in the structure of the domain itself (Olivé 2007). For example, the Unified Modelling Language (UML) (Rumbaugh et al. 2004) formally classifies its modelling constructs as either structural or behavioural: Structural semantics bring information about entities in the modelled domain, which may be true at a specific point in time, while behavioural semantics make statements about how entities in the domain change over time. (Object Management Group 2015). Also, the development of SOA and BPM systems is well served by model-driven engineering approaches, with considerable research being done in this field (Ameller et al. 2015).

The Model-Driven Architecture (MDA) framework, proposed by the Object Management Group (OMG), distinguishes between different levels of modelling abstraction for a software system. The development process moves from more abstract, technology-independent to detailed technology-specific models, eventually leading to source code or executables.

The Model-Driven Service Engineering Architecture (MDSEA) framework aims for a holistic solution to the service system design problem by applying concepts from MDA (Chen et al. 2012; Ducq et al. 2012; Agostinho et al. 2014). As such, MDSEA distinguishes three modelling levels, analogous to the MDA modelling levels:

- Business Service Models (BSM) describe the business functions, concepts, entities and attributes of the service system, such as "service", "product", "stakeholders", "organisation" and "performance indicator". The BSM uses techniques and concepts from the field of Enterprise Modelling such as the "Graphs with Results and Actions Inter-related" (GRAI) modelling approach (Chen et al. 2012; Aguilar-Savén 2004).
- Technology-Independent Models (TIM) describe the implementation of the service system at an abstract level i.e. without specific implementation details. At the TIM stage, modelling is separated in three parallel tracks, each representing a complementary aspect of the service system:
  - Information Technology: The information system and IT processes implementing the services.
  - Organisation/Human Resources: The personnel and business processes implementing the service system, such as roles and generic procedures.
  - • Physical Means: The physical infrastructures required, such as the necessary buildings and equipment.
- Technology-Specific Models (TSM) are service system models in the three afore-mentioned tracks, with additional implementation information. For example, organisational models may now include specific personnel and business rules. Physical means models may describe specific configurations and models of equipment. In the IT track, the information system models are now adapted to the target technologies which will be used to implement the service system.



المنارة للاستشارات

The IT track corresponds neatly to the problem of developing software for service systems. The final stages of the IT track involve the transformation of TIM-level models of the IT components of the service system into TSM-level models and executables, and are analogous to the PIM-PSM-executable stages of MDA. This means that MDA-based software engineering processes can be applied to build service software within the MDSEA framework, as part of a unified service engineering process.

Previous work on MDSEA suggested the use of complementary "structural" and "behavioural" modelling approaches for MDSEA's IT-track TIM level. When the service system is described in "structural" terms (i.e. stakeholders, components, data and other elements, and their relations), these can be mapped to IT-domain entities, such as interfaces, data structures and object-oriented software. On the other hand, when the service system is modelled in "behavioural" terms (i.e. inputs, outputs, processes, decision branches and others), it's possible to create software directly implementing this behaviour, like business process scripts, service compositions and applications. Specifically, Ducq et al. (2012) suggested the use of structural diagrams from the Unified Modelling Language (UML) for modelling the service system's enterprise applications and their associated data, and BPMN (Business Process Model and Notation) (White 2008) for describing its processes. For the TSM level, they proposed the use of UML diagrams for applications and data and BPEL for business processes (to be executed by a compatible BPM engine in the service system's IT infrastructure). More recently, Ducq et al. (2014) focused on BPMN 2.0 as the TIM-level output of the process, including IT aspects, citing its interoperability with BPM platforms, and ability to represent human and technical resources. Finally, when considering a practical software tool for service engineering using MDSEA, both modelling approaches have been proposed for the TIM-level: UML Class diagrams for IT artefacts and BPMN 2.0 for detailed business processes (Bazoun et al. 2014; Boyé and Bazoun 2014).

MDSEA presents several advantages as a starting point for software development for servitised manufacturing. Being derived from MDA, relevant concepts, tools and processes are applicable in the IT track. It is unique in considering software development in the larger context of complex service systems, alongside with organisational and technical aspects. The MDSEA framework does not constrain the methods and tools which can be used to implement it. This facilitates adaptations to various specific applications and domains, as well as the use of structural and behavioural modelling. The framework encourages the alignment of business goals and processes by modelling the service system as a whole in the initial design stages which then separates into organisational, physical and IT tracks. Finally, business, technical and other stakeholders participate in the design process, through common modelling abstractions.

### 2.6 Requirements for methods and tools for software development for service provision in manufacturing

Based on the information in the previous sections, we can begin to draw conclusions about the business and technological environment of software engineering for service provision. Such activities will take place in the context of the development of a larger, multi-faceted system. Information about the application domain,

المنارة للاستشارات

business processes and the overall solution must be clearly defined and communicated to the development team by domain experts. SOA and BPM technologies will be applied, and a model-driven approach (such as MDSEA) would facilitate inter-team collaboration, while maintaining consistency across platforms (Francese et al. 2015). The approach will need to combine both structural and behavioural aspects. Finally, these tools will need to be integrated and, very likely, interoperable with the IT infrastructure of the service provision system, extending the benefits of tool integration to testing, staging and deployment of applications.

These conclusions can be distilled into five requirements for a software engineering methodology and toolset aimed at product-service systems (Table 1).

### 2.7 Model-driven software development methodologies for SOA and BPM with tool support

The fields of MDE, SOA, BPM and Tool Integration may contribute towards defining a suitable process and designing the necessary tools. For this reason, we review relevant research efforts which combine all four elements to provide methodological and tool support for application development.

Work within the Eclipse framework (Eclipse Foundation 2005) has resulted in several tools and open source solutions for model-driven software engineering, SOA, and BPM. State-of-the art, general-purpose commercial tools include IBM's Rational Software Architect (Leroux et al. 2006), Sparx Enterprise Architect (Sparx Systems 2016) and others. All these environments are geared towards the general development of software systems, and several of the tools reviewed here use them as their technology foundation.

In the relevant research literature, most approaches and their tools employ either a structural or a behavioural modelling style, with very few exceptions. MDA-based behavioural modelling approaches include several methodologies and their associated toolsets. The Sensoria Development Approach (SDA) and corresponding Sensoria Development Environment (SDE) support the creation of SOA systems (Wirsing

**Table 1** Requirements for a model-driven methodology and development environment for servitised manufacturing

| |
|---|
| 1. Servitisation requires the development of a wide range of service provision applications (e.g. accounting, data collection, or web pages), therefore the necessary tools should not be limited to a single application domain |
| 2. Modelling a service system benefits from using the complementary approaches of structural and behavioural modelling for software production, so both approaches should be supported |
| 3. Integrated development environments for software development should be preferred over discrete tool chains, as the former improve productivity and are easier to manage |
| 4. Since (MDA-based) MDSEA is the only model-driven service-system development methodology with concrete support for software engineering, candidate approaches should be compatible with it by implementing MDA transformations |
| 5. The usefulness of modelling and development environments for service systems is greatly increased if they have some type of integration with the service system runtime environment and can deploy applications quickly |



المنارة للاستشارات

et al. 2008). ContextServ, is a platform specialising on context-aware web services (Sheng et al. 2009). The integration of the Service-Oriented Development Method (SOD-M) (De Castro et al. 2009) and the DENEB business process execution platform (Fabra et al. 2011) resulted in a framework for the analysis, design and execution of generic business processes (Fabra et al. 2012). MoDAR (Model-driven Development of Dynamically Adaptive Service-Oriented Systems with Aspects and Rules) models business processes with a custom toolkit targeting a specialised runtime environment (Yu et al. 2015). The CHOReOS methodology is another behavioural approach, but does not directly reference the MDA framework (Autili et al. 2013). An Integrated Development and Runtime Environment (IDRE), based on Eclipse, was implemented for CHOReOS (Ben Hamida et al. 2012).

MDA-based approaches are also present in the structural modelling camp. The MPOWER research project investigated the use of model-driven methods for the development of services for home-care applications, and relied on the HL7 framework for healthcare interoperability (Walderhaug et al. 2007). Aho et al. (2009) developed an MDA-based toolchain for web services, combining web service and database development. Non-MDA approaches include the IBM Service Oriented Modelling and Architecture (SOMA) aiming for end-to-end support for a SOA model-driven development process. (Bercovici et al. 2008). Haase and Nagl (2009, 2011) proposed a model-driven methodology and tool for the creation of service-oriented architectures as middleware for the integration of heterogeneous applications.

Both structural and behavioural approaches are present in WebRatio 5, a tool using a Model-Driven approach for the design of generic Web applications (Acerbis et al. 2007, 2008). Both structural and behavioural elements appear in the course of the A-MUSE methodology model transformations (Daniele et al. 2009a, b).

Using the criteria listed in Table 1, it is possible to evaluate whether the methods and tools reviewed are consistent with the requirements for software development in servitised manufacturing. The evaluation is summarised in Table 2.

The evaluation reveals that there is no single platform which can cover all six requirements. Several are focused on specific application domains and, thus, are not flexible enough to apply to different service fields. Overall, generic platforms such as Eclipse and Rational Software Architect need extra work to set up structural/behavioural workflows, e.g. by collecting the necessary modelling and coding plugins, and setting up the necessary interfaces. However, most other approaches are focusing on either the structural or the behavioural perspective. There are some approaches that put emphasis on the modelling aspects and do not provide a development environment. Many approaches do not support MDA-style modelling workflows, and as a result, are not compatible with the MDSEA service engineering approach. Finally, while some approaches can be integrated with production environments (i.e. service system IT infrastructures), others require additional adaptation, or are simply not designed with service system integration in mind.

المنارة للاستشارات

**Table 2** Evaluation of model-driven, SOA-oriented development frameworks with tool support

| Platform | GSPA | SWS | BWS | IDE | MDSEA-C | SRI |
|---|---|---|---|---|---|---|
| Eclipse | ● | ○ | ○ | ● | ○ | ○ |
| Rational software architect | ● | ○ | ○ | ● | ○ | ● |
| Sparx | ● | ● | ● | ○ | ● | ○ |
| A-MUSE | – | ● | ● | – | ● | – |
| STRIDE | – | – | ● | ● | ○ | ● |
| MPOWER | – | ● | – | – | ● | ● |
| WebRatio 5 | – | ● | ● | ● | ○ | ● |
| SDA | ● | – | ● | ● | ● | – |
| SOMA | ● | ● | – | ● | ○ | – |
| ContextServ | – | – | ● | ● | ● | ● |
| Aho, et al. (2009) | ● | ● | – | – | ● | – |
| Haase and Nagl (2009, 2011) | – | ● | – | – | ○ | – |
| SOD-M/DENEB | ● | – | ● | ● | ● | ● |
| CHOReOS | ● | – | ● | ● | ○ | ● |
| MoDAR | ● | – | ● | ● | ● | ● |

*GSPA* Generic Service Provision Applications, *SWS* Structural Workflow Support, *BWS* Behavioural Workflow Support, *IDE* Integrated Development Environment, *MSEA-C* MDA-based: MDSEA-Compatible, *SSI* Service Runtime Integration

●: applies, ○: requires additional components and/or adaptation, –: not applicable

## 3 Research methodology

Significant part of the research presented here took place in the framework of the "MSEE: Manufacturing SErvice Ecosystem" FP7 project. The project plan encouraged an iterative research approach for its various interrelated research tracks, including the work presented here. Therefore, the work was organised in the following phases:

*Initial definition of requirements and draft architecture* based on the characteristics of software development in servitised manufacturing, the research team made a first attempt to identify the requirements and use cases of the workflow and to produce a draft architecture of the IDE.

*Initial prototype* an initial prototype of the IDE was developed, implementing an early version of the software development workflow. Preliminary testing and feedback from the project partners underlined the need for revisions to the requirements, the user workflow, the architecture and the technologies used.

*Revised requirements and architecture* based on the experience from testing the first prototype, the requirements, workflow and architecture were updated.

*Revised prototype* based on the new specifications, a finalised prototype was developed and deployed.

*Piloting and evaluation* the prototype was deployed and was tested during the project pilots, revolving around the design development of product-service



المنارة للاستشارات

systems for manufacturing. At the end of the project pilots, software engineers were asked to assess the usefulness and impact of the workflow and the prototype IDE.

## 4 Model-driven software engineering workflow for product-service systems

In this section, we present a model driven software engineering workflow, based on model-driven principles, and as an elaboration of the final stages of the MDSEA IT track.

### 4.1 Generic process

The proposed methodology is based on a TIM-to-TSM transformation in the MDSEA "IT track". The transformation is based on MDA-style "model mapping" i.e. correlating elements of a more abstract model to elements in a less abstract one. This includes both model-type (application of element-to-element rules) and model-instance (manual mapping and model enrichment) mapping (Mukerji and Miller 2003; Dickerson and Mavris 2009; Osis and Asnina 2010). The methodology uses both of these approaches, eventually leading to the creation of executable artefacts, as seen in Fig. 1.

The generic process includes the following steps:

- Start: The TIM model is received by the developer. The model has been produced in previous steps of the MDSEA process and corresponds to a high-level description of the service IT system structure or behaviour.
- The TIM model is edited by the developer.

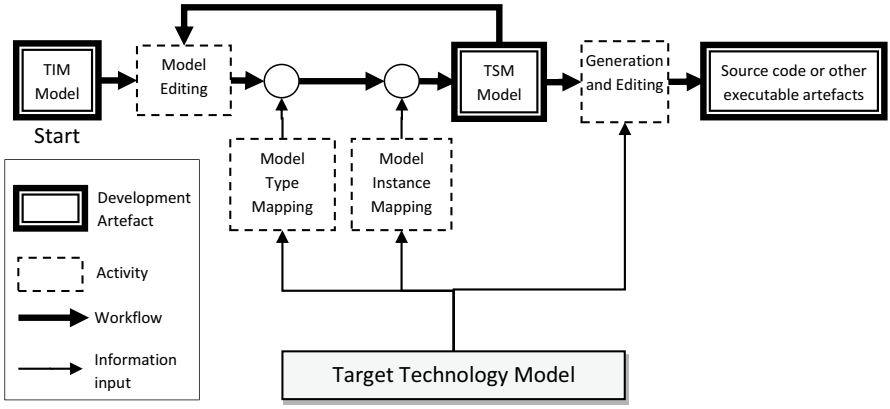


**Fig. 1** Generic TIM model to code process

المنارة للاستشارات

- Afterwards, the TIM-to TSM Model mapping process takes place using a combination of model-type and model-instance mapping. The process can be manual (performed by the developers themselves) or automated, to a degree. Both types of mapping are informed by the "Target Technology Model", i.e. the specifics of the technologies which will implement the IT system.
- The mapping process produces a TSM model based on the input TIM model.
- A generator produces code or other executables (of various levels of completeness), which may also be edited by the developers, debugged and prepared for deployment. Again, the production and preparation of executables is informed by the "Target Technology Model"
- The process results in completed executable artefacts, ready to be deployed to the target platform.

On the input side of the workflow, both UML class diagrams and BPMN 2.0 process diagrams need to be considered, based on previous work on MDSEA, and their inherent complementarity. On the output side, the resulting software is expected to operate in the framework of a SOA system with BPM elements, meaning that the process should be able to support the development of services (mainly web services) and service compositions. For added flexibility it should also support stand-alone software, development without model input (including outside of the MDSEA process), and the creation of intermediate products, such as code and models.

The generic model-driven workflow can be applied to each type of input and potential output, as described in Table 3, giving rise to a structural and a behavioural version of the generic workflow.

The combination of both approaches can support the development of complete SOA-based systems. The structural approach can be used to define specific actors or entities in the application domain, and to develop individual "atomic" services. Once these become available, developers can use the behavioural approach to model

**Table 3** Design Inputs and expected outputs

| Design input | Output |
|---|---|
| Structural descriptions: UML class diagrams | Object-oriented application code:<br>Stand-alone applications<br>Web services<br>Service compositions (with locally developed or external services) |
| Behavioural descriptions: BPMN process diagrams | Executable process scripts:<br>Stand-alone processes<br>Service compositions (with locally developed or external services) |

complex interactions between these services, and deploy these processes to service orchestration engines.

### 4.2 Structural workflow

In the structural workflow, technology-independent UML class diagrams (TIM models) are used to generate UML Class diagrams corresponding to an application in an object-oriented language (Technology-Specific Models—TSM). TIM UML class diagrams describe entities (for example documents, devices, and stakeholders) and relations in the application domain using object-oriented concepts, such as classes, components, attributes, operations, or generalisations. Despite the lack of algorithmic information (Neubauer et al. 2014), there is enough to model the basic structure of an object-oriented application.

The TSM model can be generated from the TIM model using a combination of model-type and model-instance mapping. Model-type mapping is applied to UML elements which, in turn, can be correlated with counterparts in the technology-specific domain because of their specific "type". In this case, it possible to automate this process using specific transformation rules. For example, UML classes, relationships and methods can be mapped to the corresponding Java concepts. Some technology-specific information, however, cannot be inserted in the model in this way. For example, attributes in the TIM model may be without data types, or with inappropriate ones for the target language. Model-instance mapping must be applied in this case. "Instances" of specific elements will require special treatment depending on the technology and application requirements. For example, the "price" attribute of a class will need to have a data type appropriate to the target language and the accuracy required by the end application. In this case, the developer will annotate this attribute with a specific "mark" i.e. a technology-specific semantic, indicating the proper transformation.

The result of this process is an "enriched" UML class diagram, annotated with technology-derived marks i.e. a Platform-Specific Model of the software under development, as illustrated in the example in Fig. 2. Following up from the example presented in the introduction, in this case the developer is focusing on a structural model of a part of the service system (specifically the representation of the installed equipment, i.e. heat pumps). The developer enriched the Technology-Independent model with specific java-derived semantics to prepare a Technology-Specific model targeting the Java language.

In practice, developers will further manipulate the domain model beyond this mapping process. UML structures may not correspond exactly to valid constructs in the target language: for example, Java restricts multiple inheritance to interfaces, which is not a constraint in a technology-agnostic UML diagram, while on the other hand, C++ doesn't. Beyond these mismatches, building behavioural and effective software requires adapting the model to the specific technological implementation (for example, by adding classes, methods, and fields not included in the technology-agnostic representation), re-using existing assets not referenced by the source model, and other activities.

المنارة للاستشارات

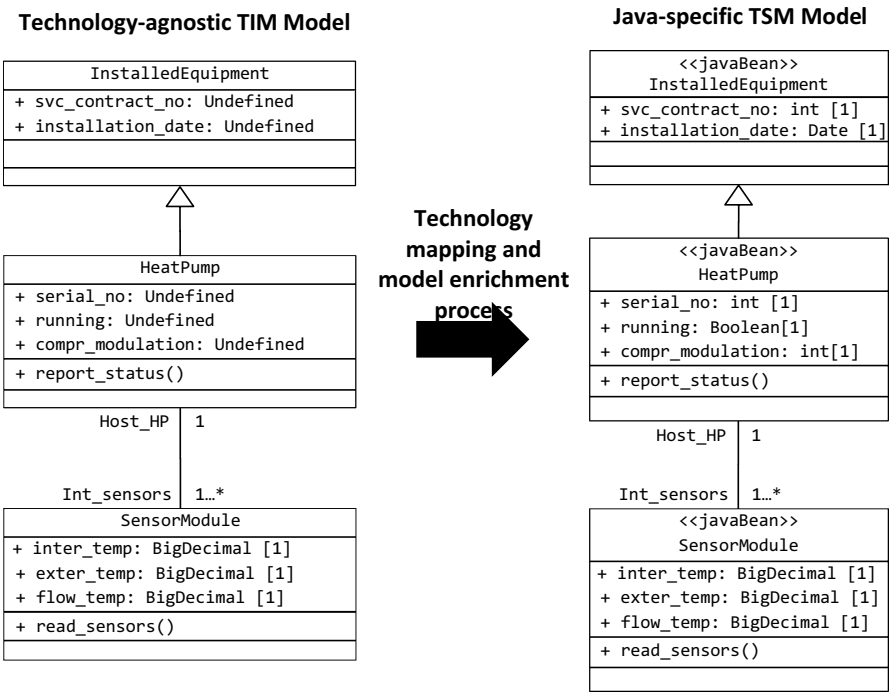


**Fig. 2** Example of TIM to Java-specific TSM mapping

```
/* HeatPump.java */
import java.util.*;
import java.math.*;
public class HeatPump extends InstalledEquipment {

private int compr_modulation;
private boolean running;
private boolean operation;
private int serial_no;


  public void report_status() {
    // Start of user code for the body of report_status
    // TODO should be implemented
    // End of user code
  }

  public int getCompr_modulation() {
    return this.compr_modulation;
  }

  public void setCompr_modulation(int newCompr_modulation) {
    this.compr_modulation = newCompr_modulation;
  }
```

**Fig. 3** Example of a Java code template generated from a TSM model



المنارة للاستشارات

By the end of this process, the TSM model will contain enough information to automatically generate a set of object-oriented code templates for all classes it contains, with appropriate variable and method declarations, and element relations, as seen in Fig. 3.

In Fig. 3, Java code has been generated for the HeatPump class, and the developer can now, for example, populate the template with code for reporting the status of the pump: the report status method. From this point on, the developers will implement software applications by populating the code templates, testing and debugging the resulting code.

### 4.3 Behavioural workflow

The generic process uses BPMN 2.0 input to produce executable scripts for Workflow Management Systems (Georgakopoulos et al. 1995), a large proportion of which can execute BPMN 2.0 process scripts (Skouradaki et al. 2015). This way, developers working strictly within the framework of BPMN 2.0, can produce executable business processes without translating them into other execution languages (e.g. BPEL) while maintaining a simple, consistent workflow.

As received from the MDSEA process, TIM BPMN 2.0 models only contain information about the business process itself, and are not adapted to any particular execution engine, (thus corresponding to MDA Platform-Independent Models). During this workflow, the users will manipulate the TIM model, in order to produce a TSM model to be executed within the framework of the manufacturing service system's IT infrastructure. The work required at this stage will depend on factors such as the business requirements, the target technology, and the maturity of the TIM model, and may include a combination of:

- Model editing: implementation and elaboration of functionality described in relatively abstract terms in the input model, improvements to workflow, usability, and consistency, the re-use of existing assets, and further BPMN elements (e.g. tasks, messaging, or flow control)
- Addition of process execution elements: These may include (among others), variables passing data during process execution, scripts, business rules and engine-specific elements such as e-mail tasks, and task assignments.
- Addition of service invocations and functionality by external systems: Services in a SOA infrastructure can be invoked during the execution of the business processes by most available BPM engines. In this case, developers include, for example, "web service tasks" which invoke, pass arguments and receive results from available services. This enables the creation of SOA-style service compositions.
- Addition of complex functionality via application code: Various engines allow developers to describe complex behaviour, which cannot be effectively captured by BPMN 2.0 semantics through the addition of additional application code.

المنارة للاستشارات

- Addition of User Interaction (UI) elements: Developers can define UI elements, such as forms and menus associated with tasks, and allow user interaction during the business process.

The result of this process is another BPMN 2.0 model of the business process containing information enabling its execution from the target BPM engine, as illustrated in Fig. 4.

Both TIM and TSM models are expressed as BPMN 2.0, with the TSM model including technology-specific extensions. Model-type mapping between the two model types is limited to the trivial correspondence of BPMN elements (e.g. tasks and events) Model-instance mapping involves the addition of various engine-derived semantics (such as tasks, workflows, and code), corresponding to an MDA PIM-to-PSM transformation.

In the example in Fig. 4, the business process involves the identification of faults in the installed equipment serviced by the manufacturer. The TIM is only concerned with high-level aspects of the process. The developer will have to enrich and update the model according to the technology infrastructure provided by the service system, i.e. the business process engine, the available web services etc. Here, the developer fleshes out the process: checking for faults means invoking the relevant web service to read the operating parameters of the existing pumps, and the application of a set of business rules for determining if a fault was found. If faults are found, the service department must be notified. The developer

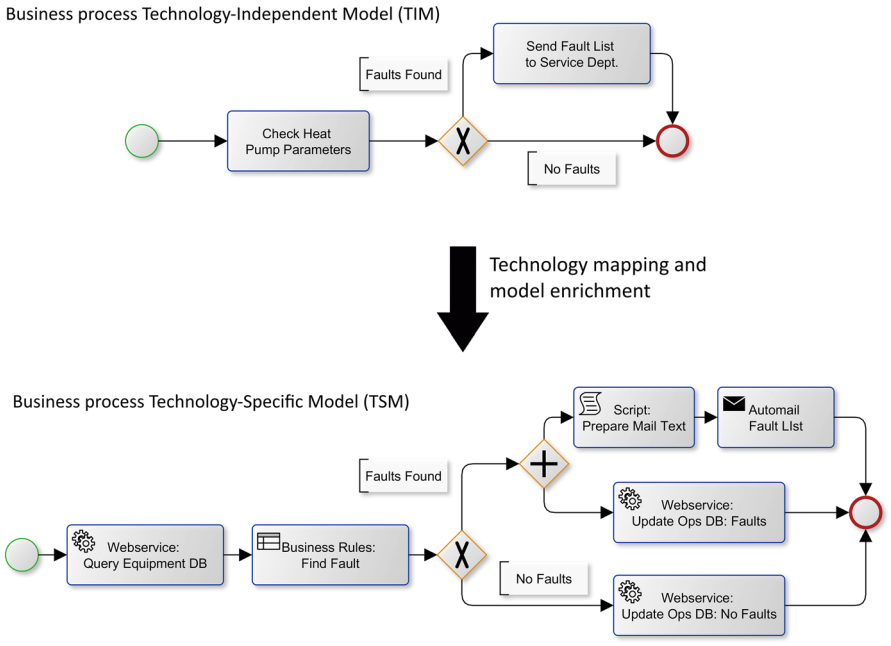


**Fig. 4** Example of BPMN TIM to TSM business process model transformation via mapping and enrichment

adds code to for the preparation of a fault list, invokes the process engine mailer to send it to the service department and updates the equipment database through an existing web service.

## 5 Development environment architecture

In this section, we present the proposed development environment architecture. First, we discuss the system context of the proposed architecture, since it defines its interfaces with other systems. Next, we describe the elements of the proposed architecture, and finally we show how these elements work together to produce the desired functionality.

### 5.1 Overview and system context

The Generic Service Development Platform (GSDP) is a model-driven development environment, supporting the workflow described in the previous section. The inputs of the GSDP are TIM (UML and BPMN) models from earlier stages of the MDSEA process. The outputs of the GSDP are applications, services, service compositions, executable business processes and intermediate products, such as models and code fragments.

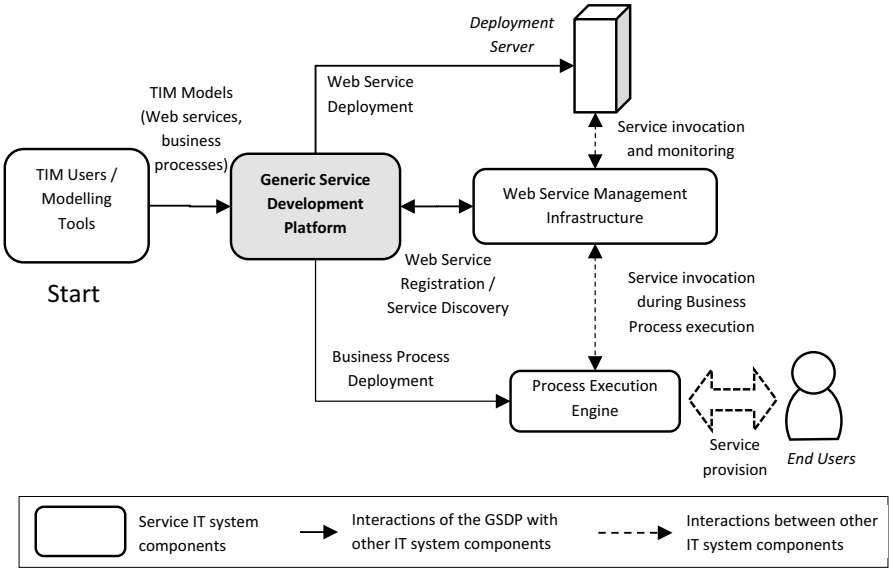


**Fig. 5** The GSDP context and interactions

As seen in Fig. 5, the GSDP exists within the IT framework of the product–service system. Specifically:

- Start: The GSDP TIM input (models for web services and business processes) come from designers using UML and BPMN modelling tools in the previous steps of the MDSEA process.
- The TIM-TSM-executable transformation takes place in the Generic Service Development Platform, as described in the previous sections.
- Developed web services are deployed to the Deployment Servers.
- When a web service is deployed to the servers, it is also registered with the Web Service Management Infrastructure of the service IT system. The registration includes metadata about the web service, the service end-point, and other information required for its invocation.
- Also, the GSDP re-uses services in the development of service compositions. This requires the discovery and retrieval of information (such as service descriptions) of available Web services. Information for service discovery is retrieved from the Web service management infrastructure of the service IT system.
- Executable business processes and service compositions are deployed to a Process Execution Engine.
- The Process Execution Engine performs the business processes and provides business services to the service system's end users (e.g. manufacturers, consumers, and network partners).
- During the execution of business processes, the Process Execution Engine invokes services registered with the Web Service Management Infrastructure. The Web Service Management Infrastructure monitors the operation of the IT system web services and operates as an intermediary between the web-service deployment servers and the Process Execution Engine.

### 5.2 GSDP IDE and model repository server architecture

Broy et al. (2010) have examined the requirements or architectural elements of an IDE supporting a model-driven workflow. According to the authors, significant issues in model driven development environments are combining different model views in a single workflow, and providing sufficient integration of the necessary tools (e.g. reducing manual transformation and input). Additionally, the necessary elements of the architecture of an "Integrated Model Engineering Environment" were briefly enumerated: (1) a common model repository, (2) model editing tools, (3) tools for model analysis and the synthesis of new artefacts (new models, code, text descriptions among others) and (4) a workflow engine guiding engineers through the development process. A common model repository has also been considered necessary by Haberl et al. (2010) for a model driven workflow and toolset. This central repository would reduce redundancy and inconsistency in collaborative development, and would support change and configuration management. Almeida et al. (2007), while examining the architectural options for



المنارة للاستشارات

a model driven service engineering environment, also emphasised the need for integration with the service provision runtime infrastructure (e.g. its execution engine or its service runtime repository).

The design of the GSDP IDE architecture is based on the extension and further elaboration of this theoretical work into a coherent architectural framework. The proposed IDE architecture supports the dual development flows (structural and behavioural) of the proposed methodology in a multi-user collaborative environment. Additionally, it foresees the inclusion of the software development tools in an expanded service engineering workflow, through interoperability with higher-level modelling tools (i.e. the design stages of the service system) and

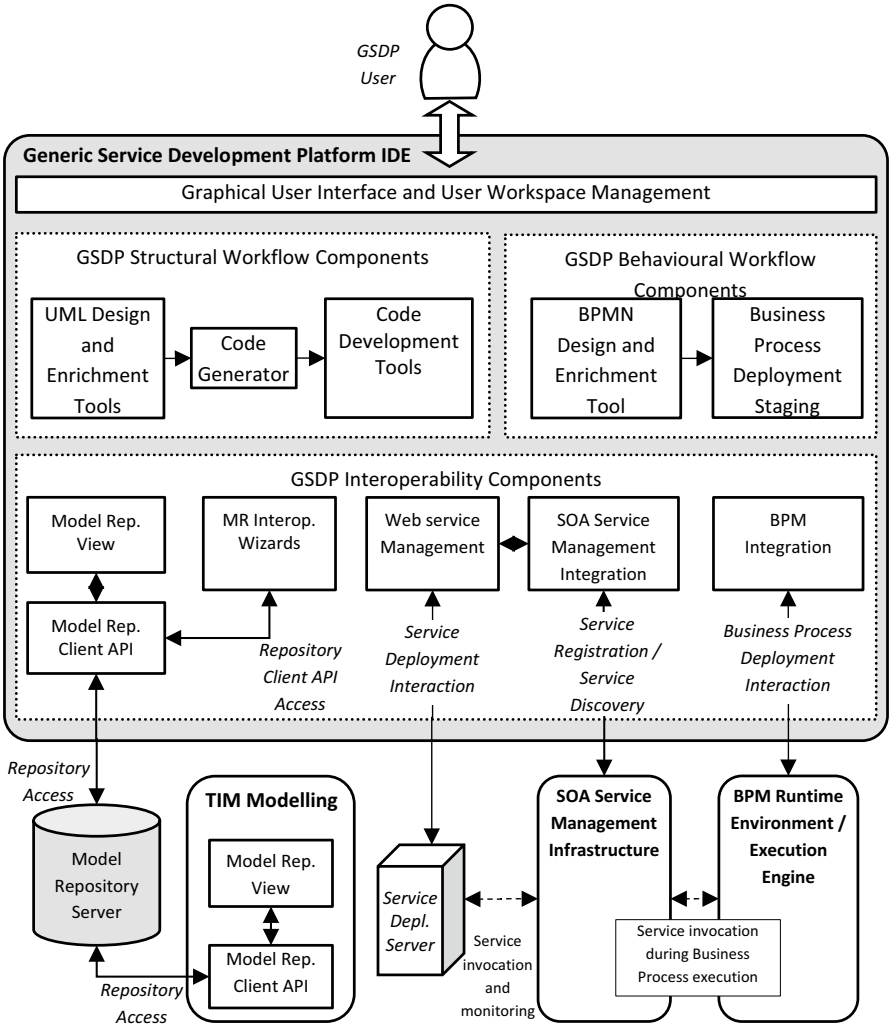


**Fig. 6** Generic Service Development Platform architecture

the runtime environment of the resulting applications (i.e. the service provision infrastructures).

The proposed GSDP IDE architecture contains the bulk of the GSDP's functionality, and is a separate instance for each user of the GSDP. It is designed as a collection of tools within a common framework providing platform, presentation, control, data and process integration, a graphical user interface and a workspace containing the tools and artefacts used in the development process (models, code, etc.). The architecture is illustrated in Fig. 6.

The GSDP architecture is composed of the following elements:

- *Core components,* which support the main functions of the GSDP IDE, i.e. model editing, mapping, enrichment, code generation and development. These are organised according to the two supported workflows: Structural and Behavioural.
- *Interoperability Components,* connecting the GSDP IDE with external systems, i.e. the Model Repository Server and elements of the manufacturing service system's IT infrastructure.
- *The Model Repository server* is a common warehouse for models and other development assets. A single MR Server instance is accessed by multiple GSDP users, as well as designers providing TIM-level models as input for the GSDP workflows (i.e. MDSEA TIM-level models of IT-track product–service system components).

#### *5.2.1 Structural workflow components*

The structural workflow components include the following:

- *UML design and enrichment tool* Used for editing and mapping technology-agnostic to technology-specific class diagrams. It includes UML editing functionality and the ability to import and apply UML profiles to models.
- *Code generator* Produces class templates from the technology-specific UML class diagram, using a set of transformation rules targeting a specific object-oriented language.
- *Code development tools* Standard tools for software development, including text editors, compilers, debuggers, and others.

The results of this process may include intermediate products (such as models and code fragments), stand-alone applications or web services which can be stored in the Model Repository.

#### *5.2.2 Behavioural workflow components*

The behavioural workflow components include the following:



المنارة للاستشارات

- *BPMN Design and enrichment tool* Used for editing BPMN process diagrams, and enriching them with technology-specific semantics. The TIM models can be imported from the Model Repository or created entirely in this environment.
- *Business Process Deployment Staging* Before being deployed to the execution engine, the business process may need to be further processed or packaged.

Intermediate products (e.g. business process diagrams, executable processes) can be exported to the Model Repository, while the finished executable processes are deployed to the BPM engine of the manufacturing service IT infrastructure.

### *5.2.3 Integration with the manufacturing service system IT framework*

A large part of the expected added value of the GSDP lies in its integration with the IT infrastructure of the manufacturing service system. Its inputs come from modelling tools used by service designers and its outputs (web services, executable business processes) are deployed in the manufacturing service system's IT infrastructures. Collaboration between stakeholders (e.g. domain experts, business analysts and software developers) requires access to common resources. Also, creating service compositions requires the discovery and referencing of available services. Finally, its finished products are deployed to their runtime environment (servers and the BPM engines running on them). This integration is supported by a set of interoperability components cited below.

*5.2.3.1 Model Repository Server and access components* The Generic Service Development platform uses the Model Repository for storing, searching and retrieving models as well as other software assets. Multiple IDE instances communicate with a central Model Repository Server through a built-in interoperability stack, as illustrated in Fig. 7.

The Model Repository functionality is contained in the following elements:

- *The Model Repository Server* Acts as a centralised "single point of truth" repository for model and software resources (e.g. code fragments, or text and documentation).
- *The Model Repository Client API* Provides an API over which other components in the GSDP IDE can communicate with the Model Repository Server.
- *The Model Repository View* The MR View component allows end users to interact with the Model Repository Server via a Graphical User Interface.
- *The Model Repository Interoperability Wizards* UI-intensive components in GSDP instances which perform various specialised operations on the Model Repository (e.g. Import, Export, and Search).

*5.2.3.2 SOA Service Management Integration* The GSDP is used to develop web services for a manufacturing service system operating on SOA principles support-

المنارة للاستشارات

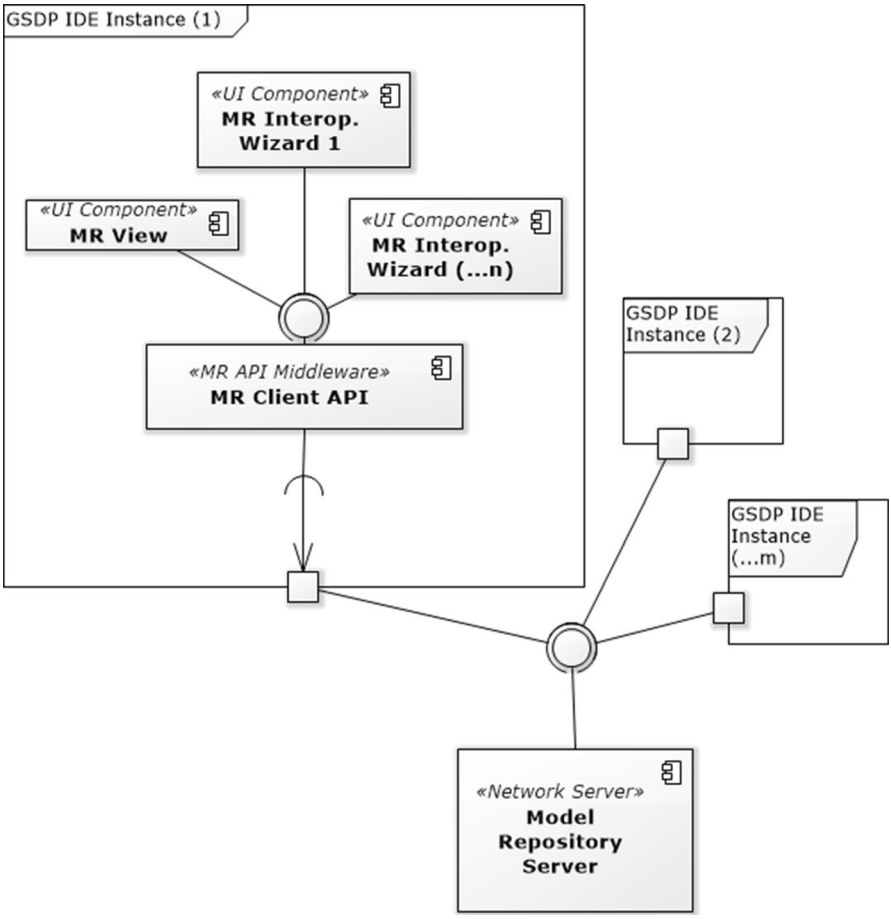


**Fig. 7** Model repository access and operations components

ing service registration and discovery. Here, communication with the GSDP is bi-directional and occurs in two cases:

- *Service registration* once a new web service is deployed to its corresponding server, the component automates the task of registering the services.
- *Service discovery* Enables searching for and retrieving information on available services (such as invocation method, parameter list and service end-point) during development.

*5.2.3.3 Web service/Business Process Management* Tasks related to web service and Business Process Management include deployment to a server, testing, generating and editing service descriptions, generating client code, and others.



المنارة للاستشارات

### 5.3 Implementing the proposed workflow

The proposed architecture is designed to support both the structural and behavioural workflows, handle user interaction and provide interoperability with the other elements of the service provision IT system. The following sections describe how the architectural components work together in the structural (web service) and behavioural (business process) workflows.

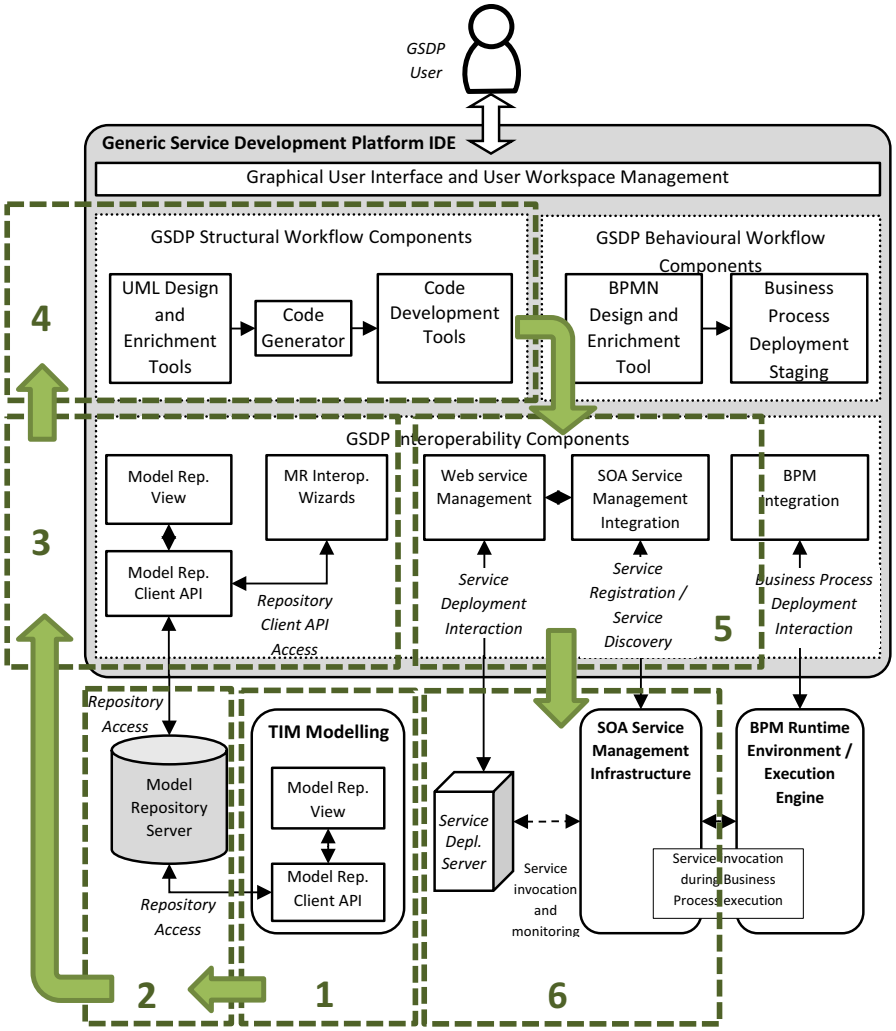


**Fig. 8** Structural workflow as implemented by the components of the GSDP architecture, organised in steps



المنارة للاستشارات

### 5.3.1 Structural workflow

In brief, the structural workflow aims to generate component web services which can be used as stand-alone services or as part of service compositions (i.e. business processes).

As illustrated in Fig. 8, the architecture components work together to implement the proposed workflow:

*Step 1* TIM structural modelling takes place outside the GSDP, by service designers, business analysts and other users. These users are provided with interoperability components for the Model Repository either as standalone tools or as part of their design environment.

*Step 2* The designers upload their TIM-level models to the common Model Repository Server. The members of the service implementation team (analysts, designers, developers and other personnel) all have access to the Model Repository, which is used as a collaboration tool and online repository.

*Step 3* The developer retrieves the models from the Model Repository. Two methods are provided, both relying on the Model Repository API: either through the Model Repository View, or Model Repository Interoperability Wizards.

*Step 4* The developer works on transforming the TIM-level model into a TSM-level model, and producing the web service code. UML tools are used to edit and enrich the model with technology-specific semantics. Code is then generated from the TSM via the Code Generator. The source code is edited, filled in, debugged and tested using conventional Code Development Tools, to produce executables artefacts (web services).

*Step 5* Once the web service is ready for deployment, the developer deploys the web service to the Deployment Server using the Web service Management component, and registers the web service with the SOA Service Management Infrastructure via the SOA Service Management Integration component.

*Step 6* The web service is deployed, registered with the SOA infrastructure and is ready for use by business processes or as a standalone service.

During the process, the developer is able to access the Model Repository Server through the Model Repository interoperability components at any time. The developer is able to retrieve and save development artefacts in the Model Repository, such as models, source code, and documents. As the Model Repository is a common resource accessible by all stakeholders (e.g. designers, developers, and analysts) it can also be used as team collaboration tool.

### 5.3.2 Behavioural workflow

The behavioural workflow aims to use TIM-level models of business processes to produce TSM-level models and, finally, executable business process descriptions. Figure 9 illustrates the architectural components used in the behavioural workflow, organised along the steps of the process.

Moving through the steps of the behavioural workflow, the components of the GSDP architecture participate as follows:



المنارة للاستشارات

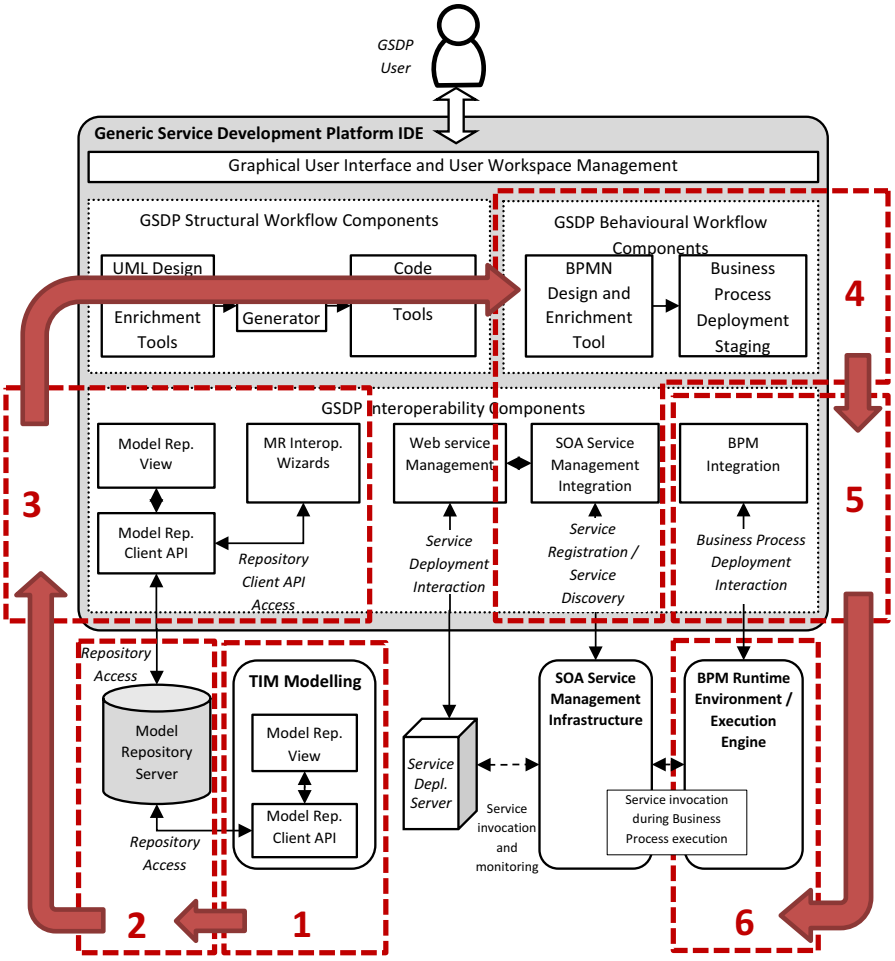


**Fig. 9** Behavioural workflow as implemented by the components of the GSDP architecture, organised in steps

*Step 1* TIM Modelling: TIM behavioural modelling for business processes takes place outside the GSDP, by service designers, business analysts and other users, as in the structural workflow.

*Step 2* The designers upload their TIM-level models to the common Model Repository Server.

*Step 3* The developer retrieves the behavioural business process models from the Model Repository, similarly to the structural workflow.

*Step 4* The developer works on transforming the TIM model into a TSM model i.e. an executable BPMN 2.0 process. The BPMN Design and Enrichment Tool is used to edit the model and enrich it with technology-specific elements. During this process, the Developer may search for suitable service components already

المنارة للاستشارات

registered with the IT infrastructure of the service system. The SOA Service Management Integration component is used to retrieve service descriptions from the SOA Service Management Infrastructure. Then, the Business Process Deployment Staging component is used to prepare the finished BPMN process for deployment to the Business Process Management (BPM) Runtime environment of the service system IT infrastructure.

*Step 5* The BPM Integration component interoperates with the BPM Runtime Environment and deploys the business process to the IT infrastructure.

*Step 6* The Business Process has been deployed and can be used by the service end users.

Again, as with the structural workflow, the developer can save and retrieve intermediate products of the behavioural workflow (models, business rules, documents, and other artefacts) in the Model Repository, and use it as a collaboration platform with other stakeholders.

### *5.3.3 Application example*

In the example of the heat pump manufacturer, the service is to be provided by a SOA-style system, powered by a BPM infrastructure. The design of the new service system is based on the MDSEA method, having produced models for the various aspects (physical, organisational and IT). The IT-track models describe mostly the entities in the service system and their relations (a structural diagram of actors and resources) and business processes performed by some or all of these entities (behavioural). The developers discuss these requirements with business-oriented personnel (from the firm, network partners or external consultants) who have provided these high-level models. The stakeholders exchange updated models and other material through the common model repository. The developers begin the development of service software using the TIM-TSM-executable process described using the integrated tools of the GSDP. The web services and other software produced this way is used support technology-specific business process orchestrations implementing the service provision. The developers deploy the web services, stand-alone software and business processes to the service system IT infrastructure, i.e. the deployment servers, the SOA service management infrastructure and the business process runtime environment. The integration with the IT infrastructure enables streamlined testing and deployment. When a competitor starts providing a similar service, business stakeholders produce a model of an updated version of the service system, which is quickly implemented, tested and put into operation.

## 6 Evaluation

The methods and tools described in the previous sections were evaluated in the context of the MSEE project. A prototype Integrated Development Environment was built and tested in a set of pilots undertaken in cooperation with industrial partners. The performance of the methodology and application was discussed with participating software engineers.



المنارة للاستشارات

### 6.1 Evaluation approach

At the pre-implementation stage, a software architecture can be evaluated using various qualitative or qualitative methods, measuring specific quality attributes. These include non-functional characteristics of the system, such as reliability, maintainability or flexibility. Qualitative approaches include "questioning" techniques where stakeholders discuss the architectures and the results are processed with a semi-formal framework (Dobrica and Niemela 2002). For example, methods such as the Software Architecture Analysis Method (SAAM) and the Performance Assessment of Software Architecture (PASA) involve the assessment of the attributes of the architecture by stakeholders with regards to specific usage scenarios (Babar and Gorton 2004). "Measuring" techniques are based on quantitative measurements of specific qualities of the architecture, employing a formal approach to calculate specific metrics. For example, object-oriented designs can be evaluated in terms of a number of formally derived metrics (such as the "coupling between object classes" or the "lack of cohesion in methods") (Chidamber and Kemerer 1994). Derived metrics can help predict attributes such as software quality (Basili and Briand 1996) or security (Chowdhury and Zulkernine 2011).

However, research in software systems can also be evaluated against its claims of contribution to the state-of-the-art (Munzner 2006). Ellis and Dix (2006) argued that the purpose of the evaluation should be to test if and when the system works or is useful for its intended goal (i.e. relevant to the problem statement). In the case of the GSDP IDE and the proposed methodology, the research is driven not by a technical but by a business concern: the successful development of Product-Service System for servitised manufacturing, through the technical facilitation of this process. Therefore, it was decided that the evaluation of the system should focus on how the new tools and way of working affected the process of software development in three areas: team collaboration, developer productivity and code quality. The topics were chosen as they are linked to the business concerns outlined in the previous sections. Team collaboration is an important element for the development of service systems, developer productivity is related to the time required to deploy an application, and code quality is linked to both the time required to test and fix bugs, as well as the quality of the end service.

The MSEE industrial pilots provided an opportunity to assess this contribution. They were focused on the strategic, business and technical aspects of conceiving, developing and implementing novel services in manufacturing Product-Service Systems. In order to support the pilot activities, the project partners would design and develop the basic elements of a service provision IT infrastructure based on SOA and BPM principles. These would include IT tools supporting the design, development and provision of services to partners and customers of servitised manufacturing firms, including the GSDP IDE and elements of its system context.

Depending on the specifics of each PSS development pilot, the GSDP IDE would be used by software engineers from the participating manufacturing firms or from other project partners specializing in software development. Therefore, these stakeholders would be able to provide an assessment of the contribution of

المنارة للاستشارات

the proposed methodology and tools, based on their experiences from using them, and their knowledge of the manufacturing or software engineering domain.

It was decided to elicit this assessment through semi-structured interviews. This approach was chosen as there are inherent difficulties in evaluating IDE usability: there is significant complexity of interaction between user traits, systems and application domains (Budgen and Thomson 2003). Standardised questionnaires, in this context, would not be able to reflect these complex interactions in a meaningful way, contributing to the challenges in IDE evaluation (Kline and Seffah 2005). This is further complicated by the fact that the GSDP IDE introduces new methods and interactions with other stages of the service provision workflow with potentially unanticipated effects. In order to understand the impact of an information system to an organisation and its stakeholders, we need to take into account the complexity of interactions in real-world settings. In these situations, interpretative and qualitative research methods may provide new insights such as highlighting issues and variables not anticipated by the researchers (Galliers 1990).

### 6.2 Prototype Integrated Development Environment

The GSDP IDE prototype environment was implemented according to the reference architecture described in the previous sections. The main IDE was based on the Eclipse Juno SR2 platform (v.4.2.2), using existing Eclipse functionality whenever available, (e.g. Java development, GUI, and user workspace management), as well as a set of specialized plugins (UML and BPMN editing, Code generation, etc.). The plugins were either custom-built for the prototype or based on existing open-source software (Table 4). Additional integration work ensured that the IDE supported the proposed workflows, from modelling to executables, within a single development environment.

**Table 4** Components used for the implementation of GSDP IDE functionalities

| Functionality | Underlying technology |
|---|---|
| Model editing (UML) | Papyrus UML (Lanusse et al. 2009) |
| Model editing (BPMN) | Activiti designer (Rademakers 2012) |
| Model enrichment (UML) | Custom UML Java profile (adapted from Papyrus UML) |
| Model enrichment (BPMN) | Java and business archive (.BAR) support (provided by Activiti) |
| Referencing external services | Eclipse Webtools (Dai et al. 2007) |
| Automatic code generation (TSM-to-code) | Acceleo plugin (Eclipse Foundation 2006) with custom transformation profile in the MOF Model-to-text Language (Object Management Group 2008) |
| Automatic code generation (WSDL-to-code) | Eclipse Webtools (Dai et al. 2007) |
| Service registration and discovery | Custom service registration and discovery wizards |
| Model repository access | Custom import wizard and graphical, point-and-click model repository view |
| Business process deployment | Custom business process deployment wizard |



المنارة للاستشارات

The Model Repository was developed as a separate network resource, commonly accessible as a server by all GSDP users. Software developers, technology designers and business analysts had access used the MR for exchanging and storing development artefacts (e.g. models and code). The Model Repository functionality was based on the WebDAV protocol (Whitehead and Wiggins 1998).

Relevant to the proposed software development process and IDE, both "input" and "output" systems were implemented by the project partners. Specifically, the service system structure and behaviour were to be modelled by business analysts in UML and BPMN using a specialised modelling tool based on MDSEA (Bazoun et al. 2014; Boyé and Bazoun 2014), to be provided to developers as input for the workflow supported by the prototype GSDP IDE. Web services developed in the GSDP were to be registered with a SOA service delivery platform of the IT infrastructure. The delivery infrastructure also provided functionality for the discovery of services during development activities (Toma et al. 2014).

### 6.3 Evaluation process

The technical assets developed as a result of research in the MSEE project were deployed in its four pilot applications of manufacturing product-service systems. The prototype GSDP IDE was applied in three of those. In two cases, the GSDP IDE was tested in parallel with traditional methods in the development of service applications. These included software for Smart TV sets and for on demand, personalised, garment manufacturing. For each of those cases, the GSDP IDE was tested in the creation of web services, service compositions and user interface elements. In the third case, the GSDP IDE was used as part of a prototype "Mobile Development Platform" for service provision through mobile devices. In this case the Mobile Development Platform was used to implement the mobile device component of a remote monitoring application for washing machines.

Nine software engineers from the project partners were asked to evaluate the contribution of the prototype platform in a set of semi-structured interviews. Individuals that participated in the design or development of the proposed methodology and the GSDP IDE were excluded from the study. The participants were asked about their experiences and opinions of the GSDP via telephone or web teleconferencing. Interviews were one-on-one, and lasted about thirty minutes each. At the beginning of the discussion, the participants were informed of the general scope of the study (i.e. "An assessment of the GSDP as a tool for service system design in manufacturing"). Then, the experts were asked three open-ended questions:

- Q1: "How did the GSDP IDE affect collaboration and during service development?"
- Q2: "How did the GSDP IDE affect in developer productivity and development time?"
- Q3: "How did the GSDP IDE affect the quality of the code produced?"

المنارة للاستشارات

The interviewer provided minimal direction during the interview (e.g. clarifications or steering the discussion back to the topic), and the participants were allowed to freely expand on their opinions.

### 6.4 Results

Regarding Q1, the participants commented that the model-driven workflow of the GSDP IDE had improved communication with service and technology designers, as well as other personnel involved in the implementation of the new service system. This effect was attributed to the following factors:

- Various architectural features supported better collaboration and communication. The model repository provided access to a common workspace with service designers. The runtime integration streamlined the feedback loop of software deployment, evaluation by business stakeholders, service design improvements, software modifications and re-deployment to the production environment.
- The MDSEA process formalises the requirements documentation process and their transmission for the multiple stakeholders involved, and establishes a cooperation framework by using common conventions (modelling, language, terminology).
- TIM and PSM models are expressed in the same language (UML, BPMN) at all levels. Modellers at the TIM level can understand the proposed PSM models while developers can point out deficiencies of the concepts in the TIM models. Both can contribute meaningfully to the finalised software design.
- Both structural and behavioural elements of the service system are modelled within the same process and toolset. Service designers and software developers can, therefore, work with a more complete picture of the service system.
- Previous stages of the MDSEA process consider several aspects of the service system, including strategic, operational and service governance issues. Despite focusing on the IT aspects of the service system, the design input for GSDP users is a result of this "holistic" approach. This contributes to better alignment between business goals and software products.
- Finally, the use of formal modelling supports the documentation and traceability of the development process. TIM and PSM models are an abstracted view of the software under development, which can be easily referenced or communicated to other team members.

However, regarding Q2 and Q3, the participants noted that there were no major effects on code quality and development productivity. The improved communication of requirements did not significantly affect the number of faults. Also, actual development time was not shorter compared to traditional methods. Even with the tools provided, a significant part of development still relied on developer effort. For example, the TIM-to-TSM-executable transformation can only be automated to an extent, as each step requires the addition of further information and elaboration of



المنارة للاستشارات

the models and software assets, while filling in the generated code templates must be done manually.

## 7 Conclusion

IT is a core enabler of servitisation in manufacturing and an important element in the provision of advanced services. However, there is no research considering the context, methods and tools for software development in manufacturing product-service system applications. Responding to this gap, we have explored the context of software engineering in product-service systems, proposed a model-driven software development workflow, and relevant software development tools based on the MDSEA framework. Also, we have described an architecture for a development platform implementing this workflow, the Generic Service Development Platform (GSDP). The methodology and IDE was evaluated during in a set of pilots, in collaboration with manufacturing industry partners. Feedback was collected from software engineers, showing that the model driven workflow and toolset improved communication between designers and developers, as well as business-IT alignment.

The results indicate that the proposed methodology and tools could be beneficial when applied to the collaborative, multidisciplinary environment of software development for manufacturing product-service systems. The model-driven aspects of the methodology provide a common language for all stakeholders involved, and support the comprehensive modelling of the service system. The associated integrated tools provide a seamless development environment for all stages of the process, as well as a collaboration framework for developers, business analysts and other personnel. Finally, while the proposed methodology and toolset were developed to cover the needs of servitisation in manufacturing, they are generic enough to be adapted to servitisation in other industrial or commercial domains.

We have identified several avenues for future research, focusing on improving the proposed workflows and tools, and expanding their scope of application. First of all, structural descriptions of a domain or service system can help produce database schemas, forming the basis of adaptations to the relevant workflow (e.g. a data-oriented branch leading to SQL for creating a database) and associated tools. Additionally, a promising line of research would involve investigating ways to include a greater range of behavioural requirements as well as non-behavioural requirements. As manufacturing organisations operate within well-defined application domains, the workload involved in the manual annotation of TIM models could be reduced by re-using TSM-level artefacts (e.g. pre-enriched classes), designing specialised UML profiles or using technology-specific meta-models for specific manufacturing fields. The service metamodel introduced by the Unified Service Description Language (USDL) (Cardoso et al. 2010) could also be investigated as a framework for TSM-level models. Additionally, domain-specific metamodels, such as those derived from Health Level 7 (HL7) are compatible with the methodology presented and could be investigated during its application in different domains. Moving away from the linear "waterfall" development process assumed by both MDSEA and MDA would require investigating ways to support more iterative or agile software development



المنارة للاستشارات

approaches (e.g. through change management). Practical "Round-trip engineering" and the consistency of models across modelling levels is also a very critical research question. Finally, target technologies beyond object-orientation and BPM require exploration of different modelling languages and transformations.

**Acknowledgements** This work has been partly funded by the European Commission through the European Commission's 7th Framework Programme and the "Factories of the Future-ICT" Project "MSEE: Manufacturing SErvice Ecosystem" (Grant No. 284860). Early results have been presented in technical reports by the MSEE project (Deliverables D42.1 and D42.2 "Generic Service Development Platform specifications and architecture"), available online at cordis.europa.eu. The authors thank the MSEE project partners for their contribution to this work.

## References

Abramovici M, Filos E (2011) Industrial integration of ICT: opportunities for international. J Intell Manuf 22(5):717–724

Acerbis R, Bongio A, Brambilla M, Butti S (2007) WebRatio 5: an Eclipse-based CASE tool for engineering Web applications. In: 7th international conference, ICWE 2007 Como, Italy, July 16-20, 2007 proceedings. Springer, Berlin, pp 501–505

Acerbis R, Bongio A, Brambilla M, Butti S, Ceri S, Fraternali P (2008) Web applications design and development with WebML and Webratio 5.0. In: Paige RF, Meyer B (eds) Objects, components, models and patterns. Springer, Berlin, pp 392–411

Agostinho C, Bazoun H, Zacharewicz G, Ducq Y, Boye H, Jardim-Goncalves R (2014) Information models and transformation principles applied to servitization of manufacturing and service systems design. In: 2014 2nd international conference on model-driven engineering and software development (MODELSWARD), pp 657–665

Aguilar-Savén RS (2004) Business process modelling: review and framework. Int J Prod Econ 90(2):129–149

Aho P, Mäki M, Pakkala D, Ovaska E (2009) MDA-based tool chain for web services development. In: Proceedings of the 4th workshop on emerging web services technology. ACM, pp 11–18

Almeida JP, Iacob M-E, Jonkers H, Lankhorst M, van Leeuwen D (2007) An integrated model-driven service engineering environment. In: Doumeingts G, Müller J, Morel G, Vallespir B (eds) Enterprise interoperability. Springer, London, pp 79–89

Ameller D, Burgués X, Collell O, Costal D, Franch X, Papazoglou MP (2015) Development of service-oriented architectures using model-driven development: a mapping study. Inf Softw Technol 62:42–66

Anzböck R, Dustdar S (2005) Semi-automatic generation of web services and BPEL processes—a model-driven approach. In: van der Aalst MW, Benatallah B, Casati F, Curbera F (eds) Business process management. BPM 2005. Lecture notes in computer science, vol 3649. Springer, Berlin, pp 64–79

Aurich JC, Fuchs C, Wagenknecht C (2006) Life cycle oriented design of technical product-service systems. J Clean Prod 14(17):1480–1494

Autili M, Di Ruscio D, Di Salle A, Inverardi P, Tivoli M (2013). A model-based synthesis process for choreography realizability enforcement. In: Proceedings of fundamental approaches to software engineering, 16th international conference, FASE 2013, held as part of the European joint conferences on theory and practice of software, ETAPS 2013, Rome, Italy, March 16–24, 2013, pp 37–52

Babar MA, Gorton I (2004) Comparison of scenario-based software architecture evaluation methods. In: Software engineering conference, 2004. 11th Asia-Pacific. IEEE, pp 600–607

Bartol N (2014) Cyber supply chain security practices DNA—filling in the puzzle using a diverse set of disciplines. Technovation 34(7):354–361

Basili VR, Briand LC (1996) A validation of object-oriented design metrics as quality indicators. IEEE Trans Software Eng 22(10):751–761

Bazoun H, Zacharewicz G, Ducq Y, Boyé H (2014) SLMToolBox: an implementation of MDSEA for servitisation and enterprise interoperability. In: Mertins K, Bénaben F, Poler R, Bourrières J-P (eds) Enterprise interoperability VI. Springer, Berlin, pp 101–111



المنارة للاستشارات

Becker J, Beverungen DF, Knackstedt R (2010) The challenge of conceptual modeling for product–service systems: status-quo and perspectives for reference models and modeling languages. IseB 8(1):33–66

Ben Hamida A, Kon F, Oliva GA, Dos Santos CE, Lorre JP, Autili M, De Angelis G, Zarras A, Georgantas N, Issarny V, Bertolino A (2012) An integrated development and runtime environment for the future internet. In: Álvarez F et al (eds) The future Internet. Springer, Berlin, pp 81–92

Bercovici A, Fournier F, Wecker AJ (2008) From business architecture to SOA realization using MDD. In: Bercovici A, Fournier F, Wecker AJ (eds) Proceedings of model driven architecture—foundations and applications, 4th European conference, ECMDA-FA 2008, Berlin, Germany, June 9–13, 2008, pp 381–392

Berkovich M, Leimeister JM, Krcmar H (2009) Suitability of product development methods for hybrid products as bundles of classic products, software and service elements. In: ASME 2009—international design engineering technical conferences & computers and information in engineering conference IDETC/CIE. San Diego, USA

Berkovich M, Leimeister J, Krcmar H (2011) Requirements engineering for product service systems. Bus Inf Syst Eng 3(6):369–380

Berkovich M, Leimeister JM, Hoffmann A, Krcmar H (2014) A requirements data model for product service systems. Requirements Eng 19(2):161–186

Bikfalvi A, Lay G, Maloca S, Waser BR (2013) Servitization and networking: large-scale survey findings on product-related services. Serv Bus 7(1):61–82

Boehm M, Thomas O (2013) Looking beyond the rim of one's teacup: a multidisciplinary literature review of product-service systems in information systems, business management, and engineering & design. J Clean Prod 51:245–260

Boyé H, Bazoun H (2014) Service life-cycle management tool box. In: Wiesner S, Guglielmina C, Gusmeroli S, Doumeingts G (eds) Manufacturing service ecosystem: achievements of the European 7th framework programme FoF-ICT project MSEE: manufacturing service ecosystem (Grant No. 284860), pp 60–66

Brax S, Visintin F (2016) Meta-model of servitization: the integrative profiling approach. Ind Mark Manage. https://doi.org/10.1016/j.indmarman.2016.04.014

Broy M, Feilkas M, Herrmannsdoerfer M, Merenda S, Ratiu D (2010) Seamless model-based development: from isolated tools to integrated model engineering environments. Proc IEEE 98(4):526–545

Budgen D, Thomson M (2003) CASE tool evaluation: experiences from an empirical study. J Syst Softw 67(2):55–75

Cardoso J, Barros A, May N, Kylau U (2010) Towards a unified service description language for the internet of services: requirements and first developments. In: 2010 IEEE international conference on services computing (SCC). IEEE, pp 602–609

Cavalieri S, Pezzota G (2012) Product-service sytems engineering: state of the art and research challenges. Comput Ind 63(4):278–288

Chae BK (2014) A complexity theory approach to IT-enabled services (IESs) and service innovation: business analytics as an illustration of IES. Decis Support Syst 57:1–10

Chen D, Ducq Y, Doumeingts G, Zachariewicz G, Alix T (2012) A model driven approach for the modeling of services in virtual enterprise. In: Zelm M, Sanchis R, Poler R, Doumeingts G (eds) Enterprise interoperability: I-ESA'12 proceedings, pp 181–187

Chidamber SR, Kemerer CF (1994) A metrics suite for object oriented design. IEEE Trans Software Eng 20(6):476–493

Chowdhury I, Zulkernine M (2011) Using complexity, coupling, and cohesion metrics as early indicators of vulnerabilities. J Syst Architect 57(3):294–313

Correia A, Stokic D, Siafaka R, Scholze S (2017) Ontology for collaborative development of product service systems based on basic formal ontology. In: 2017 international conference on engineering, technology and innovation (ICE/ITMC). IEEE, pp 1214–1221

Dai N, Mandel L, Ryman A (2007) Eclipse Web tools platform: developing Java Web applications. Pearson Education, Bloemfontein

Daniele LM, Pires LF, Van Sinderen M (2009a) An MDA-based approach for behaviour modelling of context-aware mobile applications. In: Paige RF, Hartman A., Rensink A (eds) Model driven architecture-foundations and applications. Springer, Berlin, pp 206–220

Daniele LM, Silva E, Pires LF, van Sinderen M (2009b) A SOA-based platform-specific framework for context-aware mobile applications. In: Poler R, van Sinderen M, Sanchis R (eds) Enterprise interoperability, pp 25–37

المنارة للاستشارات

De Castro V, Marcos E, Wieringa R (2009) Towards a service-oriented MDA-based approach to the alignment of business processes with IT systems: from the business model to a web service composition model. Int J Coop Inf Syst 18(02):225–260
Dickerson CE, Mavris DN (2009) Architecture and principles of systems engineering. Auerbach Publications, Boca Raton
Dobrica L, Niemela E (2002) A survey on software architecture analysis methods. IEEE Trans Software Eng 28(7):638–653
Ducq Y, Chen D, Alix T (2012) Principles of servitization and definition of an architecture for model driven service system engineering. In: van Sinderen M, Johnson P, Xu X, Doumeingts G (eds) Enterprise interoperability, pp 117–128
Ducq Y, Agostinho C, Chen D, Zacharewicz G, Jardim-Goncalves R (2014) Generic methodology for service engineering based on service modelling and model transformation. In: Wiesner S, Guglielmina C, Gusmeroli S, Doumeingts G (eds) Manufacturing service ecosystem: achievements of the European 7th framework programme FoF-ICT project MSEE: manufacturing SErvice ecosystem (Grant No. 284860), pp 41–49
Eclipse Foundation (2005) Eclipse platform. Retrieved from http://www.eclipse.org. 11 Nov 2016
Eclipse Foundation (2006) Acceleo. Retrieved from https://www.eclipse.org/acceleo/. Jan 2017
Ellis G, Dix A (2006) An explorative analysis of user evaluation studies in information visualisation. In: Proceedings of the 2006 AVI workshop on BEyond time and errors: novel evaluation methods for information visualization. ACM, pp 1–7
Fabra J, Álvarez P, Bañares JA (2011) DENEB: a platform for the development and execution of interoperable dynamic Web processes. Concurr Comput Pract Exp 18:2421–2451
Fabra J, De Castro V, Álvarez P, Marcos E (2012) Automatic execution of business process models: exploiting the benefits of Model-driven Engineering approaches. J Syst Softw 85(3):607–625
Francese R, Risi M, Scanniello G, Tortora G (2015) Model-driven development for multi-platform mobile applications. In: Product-focused software process improvement, 16th international conference, PROFES 2015 proceedings. Springer, Switzerland, pp 61–67
Galliers RD (1990) Choosing appropriate information systems research approaches: a revised taxonomy. In: Proceedings of the IFIP TC8 WG8, 2
Gao J, Yao Y, Zhu VC, Sun L, Lin L (2011) Service-oriented manufacturing: a new product pattern and manufacturing paradigm. J Intell Manuf 22(3):435–446. https://doi.org/10.1007/s10845-009-0301-y
Georgakopoulos D, Hornick M, Sheth A (1995) An overview of workflow management: from process modeling to workflow automation infrastructure. Distrib Parallel Databases 3(2):119–153
Glushko RJ (2010) Seven contexts for service system design. In: Maglio PP, Kieliszewski C, Spohrer J (eds) Handbook of service science. Springer, US, pp 219–249
Goldstein SM, Johnston R, Duffy J, Rao J (2002) The service concept: the missing link in service design research? J Oper Manag 20(2):121–134
Guillén AJ, Crespo A, Macchi M, Gómez J (2016) On the role of prognostics and health management in advanced maintenance systems. Prod Plan 27(12):991–1004
Haase T, Nagl M (2009) Service-oriented architectures and tool integration. In: Proceedings of the 8th world congress of chemical engineering. Montreal, Canada
Haase T, Nagl M (2011) Application integration within an integrated design environment. Comput Chem Eng 35(4):736–747
Haberl W, Herrmannsdoerfer M, Kugele S, Tautschnig M, Wechs M (2010) Seamless model-driven development put into practice. In: 4th international symposium on leveraging applications, ISoLA 2010, Heraklion, Crete, Greece, October 18–21, 2010. Springer, Berlin, pp 18–32
Health Level Seven International (2017) Introduction to HL7 standards. http://www.hl7.org/implement/standards/. Accessed 24 Dec 2017
Huhns MN, Singh MP (2005) Service-oriented computing: key concepts and principles. IEEE Internet Comput 9(1):75–81
Hutchinson J, Rouncefield M, Whittle J (2011) Model-driven engineering practices in industry. In: 2011 33rd international conference on software engineering (ICSE). IEEE, pp 633–642
Johnson M, Mena C (2008) Supply chain management for servitised products: a multi-industry case study. Int J Prod Econ 114(1):27–39
Jonkers H, Groenewegen L, Bonsangue M, van Buuren R, Quartel DA, Lankhorst MM, Aldea A (2005) A language for enterprise modelling. In: Lankhorst MM (ed) Enterprise architecture at work. Springer, Berlin, Heidelberg, pp 83–113



المنارة للاستشارات

Kline B, Seffah A (2005) Evaluation of integrated software development environments: challenges and results from three empirical studies. Int J Hum Comput Stud 63:607–627
Lanusse A, Tanguy Y, Espinoza H, Mraidha C, Gerard S, Tessier P, Schnekenburger R, Dubois H, Terrier F (2009) Papyrus UML: an open source toolset for MDA. In: Proceedings of the fifth European conference on model-driven architecture foundations and applications (ECMDA-FA 2009), pp 1–4
Leroux D, Nally M, Hussey K (2006) Rational software architect: a tool for domain-specific modeling. IBM Syst J 45(3):555–568
Lim CH, Kim KJ (2015) IT-enabled information-intensive services. IT Prof 17(2):26–32
Lim CH, Kim MJ, Heo JY, Kim KJ (2015) Design of informatics-based services in manufacturing industries: case studies using large vehicle-related databases. J Intell Manuf. https://doi.org/10.1007/s10845-015-1123-8
Lindström J, Löfstrand M, Karlberg M, Karlsson L (2012) A development process for functional products: hardware, software, service support system and management of operation. Int J Prod Dev 16(3–4):284–303
Lopez DM, Blobel BG (2009) A development framework for semantically interoperable health information systems. Int J Med Informatics 78(2):83–103
Maglio PP, Vargo SL, Caswell N, Spohrer J (2009) The service system is the basic abstraction of service science. IseB 7(4):395–406
Martínez-Garciá A, García-García JA, Escalona MJ, Parra-Calderón CL (2015) Working with the HL7 metamodel in a model driven engineering. J Biomed Inform 57:415–424
Maussang N, Sakao T, Zwolinski P, Brissaud D (2007) A model for designing product-service systems using functional analysis and agent based model. In: International conference on engineering design, ICED'07. Paris, France
Meier H, Volker O, Funke B (2011) Industrial product-service systems (IPS). Paradigm shift by mutually determined products and services. Int J Adv Manuf Technol 52:1175–1191
Metzger D, Niemöller C, Thomas O (2017) Design and demonstration of an engineering method for service support systems. IseB 15(4):789–823
Mietinnen S, Rontti S, Kuure E, Lindström A (2012) Realizing design thinking through a service design process and an innovative prototyping laboratory—introducing Service Innovation Corner (SINCO). In: Proceedings of the conference on design research society (DRS 2012)
Mikusz M (2014) Towards an understanding of cyber-physical systems as industrial software-product-service systems. In: Procedia CIRP. Product services systems and value creation. Proceedings of the 6th CIRP conference on industrial product-service systems, vol 16, pp 385–389
Morelli N (2002) Designing product/service systems: a methodological exploration. Des Issues 18(3):3–17
Mukerji J, Miller J (2003) MDA guide version 1.0.1. The Object Management Group (OMG), Needham
Munzner T (2006) A nested model for visualization design and validation. IEEE Trans Visual Comput Graphics 15(6):921–928
Neely A (2008) Exploring the financial consequences of the servitization of manufacturing. Oper Manag Res 1(2):103–118
Neubauer P, Mayerhofer T, Gerti K (2014) Towards integrating modeling and programming languages: the case of UML and Java. GEMOC 2014, p 23
Nguyen HN, Exner K, Schnürmacher C, Rainer S (2014) Operationalizing IPS$^2$ development process: a method for realizing IPS$^2$ developments based on process-based project planning. In: Procedia CIRP 16, pp 217–222
Object Management Group (2008) MOF model to text transformation language specification. Retrieved from http://www.omg.org/spec/MOFM2T/About-MOFM2T/. Jan 2017
Object Management Group (2015) Unified modeling language, version 2.5. http://www.omg.org/spec/UML/2.5. Accessed 20 July 2017
Olivé A (2007) Conceptual modeling of information systems. Springer, Berlin
Osis J, Asnina E (eds) (2010) Model-driven domain analysis and software development: architectures and functions. IGI Global, Hershey
Papazoglou MP, Georgakopoulos D (2003, October) Service oriented computing. Commun ACM 46(10):25–28
Pernstål J, Gorschek T, Feldt R, Florén D (2015) Requirements communication and balancing in large-scale software-intensive product development. Inf Softw Technol 67:44–64
Pezzotta G, Sala R, Pirola F, Campos AR, Margarito A, Correia AT, Fotia S, Mourtzis D (2016) Definition of a PSS engineering environment: from the theoretical methodology to the platform

المنارة للاستشارات

implementation. In: XXI Summer School Francesco Turco 2016-smart manufacturing: new paradigms for a smarter world, vol 13. AIDI-Italian Association of Industrial Operations Professors, Naples, pp 97–101
Qu M, Yu S, Chen D, Chu J, Tian B (2016) State-of-the-art of design, evaluation, and operation methodologies in product service systems. Comput Ind 77:1–14
Rademakers T (2012) Activiti in action: executable business processes in BPMN 2.0. Manning Publications Co, New York
Reim W, Parida V, Örtqvist D (2015) Product-service systems (PSS) business models and tactics—a systematic literature review. J Clean Prod 97:61–75
Rosen M, Lublinsky B, Smith KT, Balcer MJ (2012) Applied SOA: service orietned architecture and design strategies. Wiley, Hoboken
Rumbaugh J, Jacobson I, Booch G (2004) Unified modeling language reference manual, the. Pearson Higher Education, Bloemfontein
Saarijärvi H, Grönroos C, Kuusela H (2014) Reverse use of customer data: implications for service-based business models. J Serv Mark 28(7):529–537
Sheng QZ, Pohlenz S, Yu J, Wong HS, Ngu AH, Maamar Z (2009).ContextServ: a platform for rapid and flexible development of context-aware web services. In: Proceedings of the 31st international conference on software engineering. IEEE Computer Society, pp 619–622
Skouradaki M, Roller DH, Leymann F, Ferme V, Pautasso C (2015) On the road to benchmarking BPMN 2.0 workflow engines. In: Proceedings of the 6th ACM/SPEC international conference on performance engineering. ACM, pp 301–304
Sparx Systems (2016) Enterprise architect. Retrieved from http://www.sparxsystems.com/. 11 Nov 2016
Stokic D, Correia AT (2015) Context sensitive Web service engineering environment for product extensions in manufacturing industry. In: 7th international conference on advanced service computing. Nice
Thomas I, Nejmeh BA (1992) Definitions of tool integration for environments. Software 9(2):29–35
Toma I, García JM, Larizgoitia I, Fensel D (2014) A semantically enabled service delivery platform: an architectural overview. In: Ramanathan R, Raja K (eds) Handbook of research on architectural trends in service-driven computing. IGI Global, Hershey, pp 181–186
Van Riel AC, Lievens A (2004) New service development in high tech sectors: a decision-making perspective. Int J Serv Ind Manag 15(1):72–101
Vandermerwe S, Rada J (1988) Servitization of business; adding value by adding services. Eur Manag J 6(4):314–324
Vargo SL, Maglio PP, Akaka MA (2008) On value and value co-creation: a service systems and service logic perspective. Eur Manag J 26(3):145–162
Vasantha G, Roy R, Lelah A, Brisaud D (2012) A review of product-service systems design methodologies. J Eng Des 23(9):635–659
Vogl GW, Weiss BA, Helu M (2016) A review of diagnostic and prognostic capabilities and best practices for manufacturing. J Intell Manuf. https://doi.org/10.1007/s10845-016-1228-8
Walderhaug S, Stav E, Mikalsen M (2007) The MPOWER tool chain-enabling rapid development of standards-based and interoperable homecare applications. In: Proceedings of Norsk Informatikk Konferanse (NIK 2007), pp 103–107
Wallin J, Parida V, Isaksson O (2015) Understanding product-service system innovation capabilities development for manufacturing companies. J Manuf Technol Manag 26(5):763–787
Wasserman AI (1990) Tool integration in software engineering environments. In: Long F (ed) Software engineering environments. Springer, Berlin, pp 137–149
White SA (2008) BPMN modeling and reference guide: understanding and using BPMN. Future Strategies Inc, Pompano Beach
Whitehead EJ, Wiggins M (1998) WebDAV: IEFT standard for collaborative authoring on the Web. IEEE Internet Comput 2(5):34–40
Wirsing M, Hölzl M, Koch N, Mayer P, Schroeder A (2008) Service engineering: the sensoria model driven approach. In: Proceedings of software engineering research, management and applications (SERA 2008), pp 20–22
Yu J, Sheng QZ, Swee JK, Han J, Liu C, Noor TH (2015) Model-driven development of adaptive web service processes with aspects and rules. J Comput Syst Sci 81(3):533–552
Zhao Z, Cai X (2013) Research on modeling framework of product service system based on model driven architecture. In: The 19th international conference on industrial engineering and engineering management. Springer, Berlin, pp 1283–1290



المنارة للاستشارات

Information Systems & e-Business Management is a copyright of Springer, 2018. All Rights Reserved.

www.manaraa.com