A Coopetition Space for Complex Product Specification

Mohammad Shafahi, Hamideh Afsarmanesh, Mahdi Sargolzaei

A Coopetition Space for Complex Product Specification

Mohammad Shafahi, Hamideh Afsarmanesh and Mahdi Sargolzaei

University of Amsterdam, Informatics Institute, FCN group,
Science Park 904, 1098 XG Amsterdam, The Netherlands
{m.shafahi, h.afsarmanesh, m.sargolzaei1}@uva.nl

Abstract. Due to peculiarities and complexities embedded in complex service-enhanced products, e.g. automated buildings, these products are one-of-a-kind, largely customized, and may involve a large number of competitive / cooperative multi-stakeholders. Life cycle of complex products typically includes a substantially long creation phase, followed by its operation and evolution phases that last over decades. Although, the majority of complex product components (e.g. equipment and services) are specified gradually and by varied stakeholders during its creation stage, further specifications are also provided later on to support its evolution. This paper addresses challenges in both specification of varied and numerous components, and managing these specifications thought-out the complex product life cycle. We address reusability, modularity, and federated sharing requirements in the coopetition space of complex product specification, and within the context of Virtual organizations Breeding Environments (VBEs). Our developed product specification system, which is already alpha tested, addresses these identified requirements, and is described and exemplified.

Keywords: Complex Products, Product Specifications, Virtual organizations Breeding Environments (VBE), Service-enhanced Products, Coopetition

1 Introduction

Complex products (e.g. solar power plant and intelligent buildings) are one of a kind in their design specification and massively customized in their production. Furthermore, the Product Life Cycle (PLC) of such complex products runs over several decades. The specification of complex products is therefore typically not performed in one session, rather iteratively during its life cycle, and potentially involving a number of different stakeholders, from equipment manufacturers and service providers, to experts at an EPC (Engineering, Procurement, and Construction) company. These stakeholders need to collaborate within a coopetitive environment, in order to gradually and incrementally specify different components and sub-components related to the complex product.

Additionally, based on our findings in the area of solar plants and intelligent buildings, which are relatively young industries, the design and engineering of these complex products cannot be resulted through the mere searching and identification of the needed components among the existing products in the market. In other words, although familiarity with the existing related product/service details, as provided by different manufacturers and suppliers in the market, are the necessary starting point for

the complex product designer, the mere existence of these product details are not sufficient to fully specify the ***Project Design*** of the complex product. Rather, the nature of our targeted complex products mandates detailed and concise design and customization processes for its sub-products, including the equipment, devices, and enhancing services, as well as involving different stakeholders in these processes.

We propose an environment for complex product specification, to provide the coopetition space needed through different PLC phases. A number of earlier research works address collaborative environments for product specification and design, e.g. for collaborative CAD systems [1][2]. In this paper however, we investigate requirements for a product specification environment that can support collaboration among competing companies (the so called Coopetition), within the context of VBEs and goal oriented Virtual Organizations (VOs). Considering the coopetition environment that is supported in the complex product VBEs, a main aim in this environment is to support the *reusability, modularity,* and *sharing* of the generated assets. As mentioned above, the addressed complex products are young industries and therefore their stakeholders can very much benefit from sharing the specification of sub-products that are designed by others. Therefore, supporting both the reusability of sub-product specifications and the possibility of granting access privileges on them to other stakeholders are important requirements. Furthermore, considering that these complex products are one-of-a-kind, their designed sub-products can be reused only in the case where sub-product's specifications follow a modular design approach, so that the pieces of their specification can be accessed and copied for reuse.

Besides the specification of various equipment and devices needed for complex products, we also address the specification of variety of needed business services, that can range from software systems to human-provided (the so called manual) services, and which in one way or another enhance the complex product.

In the knowledge-based economy, services have an increasingly important role in manufacturing industries, which use functionality provided by services to differentiate their products [3]. In fact, by adding business services, while it also increases the value of the products, a higher level of differentiation can be realized [4]. Therefore, in our design of the specification framework for complex products, we consider that sub-products typically come with a set of business services that offer some beneficial enhancement to the customers of these products. Capturing different aspects of these business services as well as the inter-relationships and links between these services and other sub-products of the complex product (e.g. devices), are main requirements for our proposed complex product specification framework and system. Many approaches and standards have been developed by the research community in the area of Service Oriented Architecture (SOA) to specify and formalize business services [5]. There are however still challenging and open questions related to how make services interoperable, so that they can be shared and reused, as well as how to assist authorized service providers with composing other services, thus producing value-added service to support complex products. Furthermore, there are still gaps in correlation between services and products in the context of complex products.

Please note that this research on product specification framework is performed within the GloNet[1] [6] project, and constitutes one of its subsystems.

[1] http://www.glonet-fines.eu

The remaining sections of this paper are structured as follows. In Section 2 we address the role of product specification in different phases of our target complex products' life cycle. In Sections 3 and 4 we will focus on how to realize the main requirements (non-functional and functional) for complex product specification framework and provide some details related to the implementation. Finally in Section 5, some concluding remarks are provided.

2 Product Specification in Different Phases of the PLC

The PLC of a complex product can be divided into the following three main phases [7], each having its own peculiar features:

(i) **design and engineering,**
(ii) **construction and commissioning,** and
(iii) long term **operation and maintenance**.

In this section we address these three phases, primarily in relation to the specific product specification needs of stakeholders that are involved in the phases. But before focusing on each phase, we should point out that the product specification is also needed before the PLC of the complex product starts. This is mainly due to the need for preparation of the bid for the targeted complex product, for instance in response to a call for tender. Figure 1 indicates the product specification process during different phases of the complex product's PLC, as well as during the pre-phase of bidding for the complex product, in order to preform cost estimation and initial partner selection.

Design and Engineering phase: Being the first phase in the PLC of the complex products, the design and engineering phase plays an important role in the success of the later phases. Activities during this phase are typically divided into the three steps of: project assessment, project design, and project implementation. *Project Assessment step* includes the complete analysis of the site and the technical assessment of the entire project, at this step a ***high-level specification of the complex product*** is made in order to assess the feasibility of the project. After the Project Assessment has been successfully preformed, during the *Project Design step* the early engineering and selection of technology takes place. These include the following activities, which are reflected in some ***detailed specification of complex product***:

(i) Pre-engineering - e.g. achieving initial specifications of the complex-product,

(ii) Evaluation/selection of technology or equipment - e.g. evaluating for selection or extension of existing devices and equipment suitable for complex product), and

(iii) Selection of sub-product specifications - e.g. adding sub-product specifications as components of complex- product specification.

Finally after the Project design step has been successfully preformed, during the *Project Implementation step*, the ***planned specification of the complex product is finalized*** and the product specification is used for selecting the relevant organizations and for sub-product procurement.

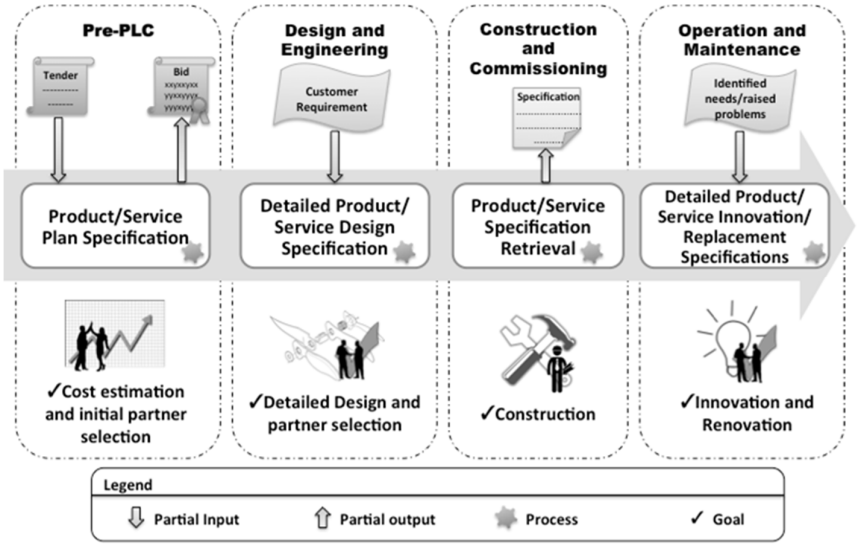

Figure 1- Product specification in different phases of complex product's PLC

Construction and Commissioning phase: Typically during this phase, no new product specification is made; rather the existing ones might be accessed to retrieve some detailed design information.

Operation and Maintenance phase: Although as discussed above, the product specification is fundamentally required during the entire design and engineering phase of the product life cycle, it is also usually needed to be used later during the long-term operation and maintenance, but rather infrequently. This occurs mostly due to the continuous need for evolution of the complex product and/or to innovate and provide new products either in response to "newly identified" needs or some problems emerged during the operation phase. Example cases of such requirements vary from enhancement or upgrading a control box in a solar plant, to replacing the panels damaged in an earthquake.

3 Realization of Non-functional Requirements

At its base, system requirements address *why* a system is needed, *what* are the functions it must provide, *how* the system must be constructed and implemented, and *what* conditions must be satisfied by the system. The two main types of requirements are the *non-functional* and the *functional* requirements [8]. This section addresses the most important non-functional requirements for our product specification space, namely the: *security, integrity, scalability,* and *portability*.

3.1 Security

Security plays a very important role in systems. This is due to the fact that improper access to a system might bring loss and even bankruptcy to the organization using the

system. Proper prevention of threats via competitors, both from the outside world (enforcing authentication) and from the inside of the system (enforcing authorization), is a must. The main steps in this process are addressed below.

3.1.1 Authentication

There are three different main techniques that can be used for authentication, including [9]: *what you are, what you have,* and *what you know.*

Among the above, and considering the usage/user of the product specification space, we have selected the *"what you know"* technique. In this approach the product specification sub-system (developed on top of the cloud-based GLONET platform) receives a *token* about each user's authenticity. This token is generated through the "Single Sign-On" mechanism implemented within in the GloNet platform [10].

3.1.2 Authorization

A very important requirement for any system that deals with multiple stakeholders (specifically within a coopetitive environment) is its secure and proper information sharing as well as mechanisms for granting access privileges to authorize users. This is due to the fact that although different stakeholders in VBEs may cooperate to achieve some specific common goals, they are potential competitors on many others.

To preserve users of the product specification space, against unwanted data access to their private information, we have designed and implemented three different data spaces (levels of access) for users of this system, that limit who can access what. As also illustrated in figure 2, these spaces include: Private – only for personal use of the user, Restricted – specified by the user to be shared only with the partners of certain VO or one specific project, and Public – to be shared with all in the VBE.

Please note that the user who defines a product specification within the system is the owner of that specification. Therefore, only that user can with some condition move such specification from one data space to another, e.g. from private either to restricted or public, in order to share it with others.

Private data space: The first step in the process of product specification is to specify sub-products of the complex product. To accomplish this task, designers should have a private space to do their specifications before making them available to other stakeholders involved in specification process of the complex product. This space is called *private space* and specifications in this space are only accessible by its owner.

Public data space: After one has specified a product he/she might wish to share the specification with the public, meaning within the VBE. This can be exemplified by a sub-product manufacturer or provider who wishes to promote the use of its already built product (e.g. an equipment), to be used as a sub-product for building other complex products. However, a number of users nowadays may be involved and interested to participate in open access movements. To enable this possibility, system provides a VBE public space for users, to provide access to owned specifications.

Restricted data spaces: Within the process of specifying one complex product, multiple stakeholders are typically involved. This usually happens within a VO, when users would be interested to share certain specifications only with the other VO partners. To enable this feature, one user can indicate the restricted space for its product

specification. Consequently, every time a VO is created a restricted data space is created for it.

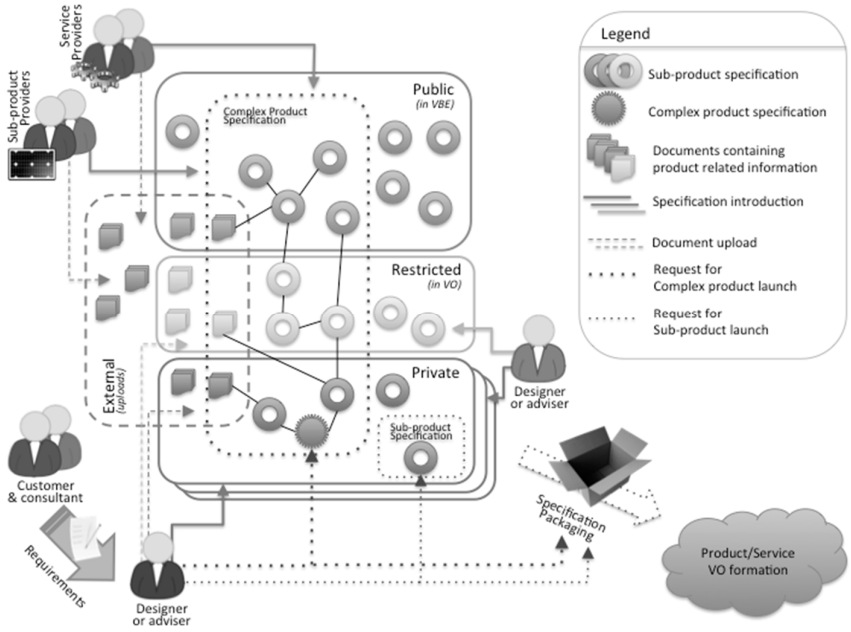

Figure 2 – Three Data spaces in relation to product specification framework

We shall now address how the users can view and/or move their owned specifications from one data space to other data spaces. Once a product is specified, it can be viewed by its owner. The screenshot in Figure 3 illustrates how product specifications can be viewed by a user. Depending on the selected *VO-name* or *project-name* (as indicated in the upper right corner of the screen), a list of associated specifications for which the user is authorized to view will appear, sorted by their product names. Please note the symbols that appear in front of the product names, where (-) represents private and (#) represents restricted. Also note that the semantics of *projects* and *VOs* are very different, and while the former indicates one optional user-defined folder, the latter is dedicated to all restricted specifications belonging to a specific VO. Please also note the following three cases:

- If neither a specific project nor a specific VO is identified by the user (on top right of the screen), then all ***public*** product specifications in the system, further to all ***private*** specifications of that user, will be illustrated.
- If no project is mentioned by the user, but a VO is specified for which the user is authorized, then only ***restricted*** specifications related to that VO will be illustrated.
- If the user specifies no VO, but a project-name, then all the ***private***, ***restricted***, and ***public*** product specifications that the user has associated to that folder will be shown.

In the example of Figure 3, the user Prolon has selected/indicated the VO's name "Electrical Design". Consequently, all restricted product specifications that belong to

this VO, as well as the all those public sub-products associated to this VO are illustrated. Please note that while Prolon might itself own some of these restricted products, other users (who are also partners of this VO) own the others.

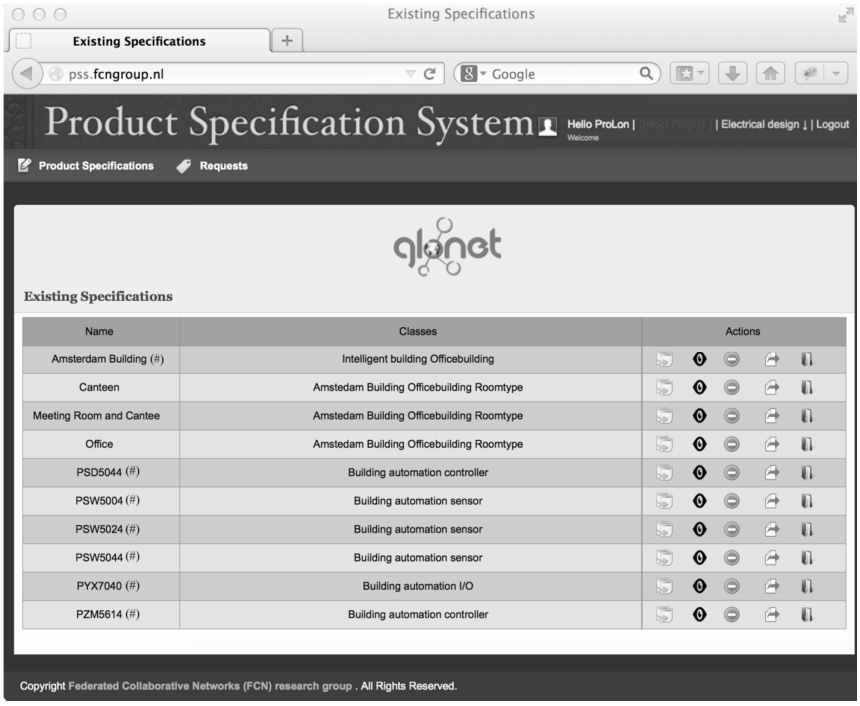

Figure 3 – Existing Specifications Window (restricted to one VO)

Other than viewing the product specifications, authorized users can also change the accessibility of these specifications by preforming the following set of actions:

- Share action (shown with icon), which provides the user with the option to change the access rights/sharing status of a certain product specification that he/she owns. The share options are available through existing products window, when the user clicks on its icon. Please note that when defining a new product specification, by default the access right to that specification is made private, that is if the user has not indicated a VO on the top right corner of the screen, in which case by default the specification becomes restricted to that VO. At any point in time, the owner of the product specification is allowed to only broaden the access to that specification. This means that if a specification is private, then the owner can change it either to restricted within a VO, or to public. In other words once the owner of a product specification grants certain access rights to others (e.g. to view the specification) he/she cannot withdraw that right later.

- Assign to Project action (shown with icon), which provides the possibility to provide access to an already defined specification, to which user has access and is

already indicated as private, public, or restricted, to an existing project folder of that user. This is mainly to assist the user with organizing his/her product specifications in different folders. This means that by default if a project is indicated on top right corner while specifying a product, then that product will be allocated to that project folder. Nevertheless, through this action, as provided in the Existing Specification window, specifications may be assigned and reassigned to different project folders.

- Delete action (shown with icon ⊖), which allows hiding certain specification(s) from the users' screen, for instance if the user finds a sub-product useless for him/her to keep, its specification can be deleted from his/her view.

3.2. Scalability

During its PLC, a complex product might potentially deal with hundreds of users. To support such a user base, and to enable possible expansions to both the user base and the product specifications, we have leveraged the possibilities supported by the cloud environment that allows allocation of more resources on demand, when and if needed. This leveraging has been done through applying three different techniques. *First*, different components of the system (i.e. the executable building blocks of mainly the Controller layer and the DAO implementation layer) are decoupled from each other, which include decoupling: the web service controllers, web interface controllers, Hibernate DAO Implementations, GloNet DAO Implementations, etc. This means that none of these components depends on how another component is implemented or executed, which in turn enables the execution of different components on different physical and/or virtual machines. *Second*, the implemented product specification space is layer-based (e.g. having Application, Data, etc. layers), while existence of each layer is transparent to the other layers. Consequently, not only the components of our designed system are decoupled, but also the different layers are decoupled and can run on different physical/virtual machines. And *third*, the implemented system can take advantage of load balancing mechanisms for their data, supported through separation of data access to objects, and how it is implemented.

3.3 Portability

The product specification space has been developed as a web-based application, using Java programming language that enables the server side code to run independent of the platform. The server side program generates standard outputs (e.g. HTML 5 [11]) that could be consistently and easily rendered by different browsers. The client side (Browser side) of the program is based on JavaScript and is written using jQuery framework [12], to insure that JavaScript code is compatible with different browsers.

The combination of java as the programming language of the server side and the compatibilities of the client side, such as compatible JavaScript and standard HTML 5, makes both the client (browser) and the server side of the system highly portable.

4 Realization of Functional Requirements

The *complex product specification space*, addressed in this paper, needs to provide a set of functionalities to support: ***product and service specification and registration.***

Within the context of the *data level* of the PLM (Product Lifecycle Management) framework [13] for complex products, three main product models need to be captured throughout its lifecycle. These include: its Geometric-oriented product models (e.g. through CAD system), the Structure-oriented product models (e.g. through DMS – Digital Manufacturing System), and the Meta-data-oriented product models (e.g. through a database). The *product and service specification space* primarily captures, handles, and manages the detailed *meta-data* about the complex product and all its sub-products, and assists users with their specification. Furthermore, the product specification space captures and stores links to a set of files that represent the other two product models, which are mainly produced in certain industry specific systems, e.g. CAD, CAM, and DMS software.

The main requirements for the functionality provided by the product and service specification space are three-fold. The **first** requirement is to support ***gradual*** specification of the complex products. This is needed to reflect the reality of complex products that are neither defined in one session, nor by one stakeholder. Therefore, detailed specifications that capture and transform customer requirements for a complex product into discrete sub-product specifications, can be gradually defined by the involved multi-stakeholders, using the developed product specification space. The **second** requirement is to properly capture the classification of all relevant sub-products in a ***granular and modular*** manner in the complex product environment, e.g. distinguishing and capturing both the electrical and mechanical aspects of a sub-product, as well as their inter-relationships. This will in turn support effective multi-perspective retrieval/discovery of information related to sub-products, as well as creating their concise descriptions, as needed for common understanding among different related stakeholders. The **third** requirement is to capture all details related to sub-products in a ***reusable*** from. As such, the existing specifications of already introduced sub-products can be either fully or partially (e.g. at the level of certain detailed feature-kind) reused for the specification of other sub-products.

In the following subsections we address an approach and different steps involved in defining details of sub-products and services related to complex products, through the use of the *product and service specification and registration space*.

4.1 Supporting granular and customized specification of sub-products

Supporting different levels of granularity and customizability is a necessity for complex products, due to their dynamic and complex nature. At the lowest level of granularity, the *features* of a specific sub-product can be defined. Every feature is an instance of a *feature-kind*. Through the granular definition of feature-kinds and instantiating the features, the system enables the user to specify any sub-product from scratch, and without being limited to only defining sub-products as instances of already existing type of products, with a pre-defined set of fields/attributes.

Here the required functionality for the product specification space includes enabling the user to define feature-kinds, as needed for definition of classes of sub-products, as well as to specify the sub-products based on providing their features. Furthermore, the specification of feature-kinds makes them reusable, so that once they are defined; all users can use them both for the specification of new class of sub-products, as well as

for instantiating features related to specific sub-products. Example screen shots of the product specification space are presented in Figure 3 and Figure 4.

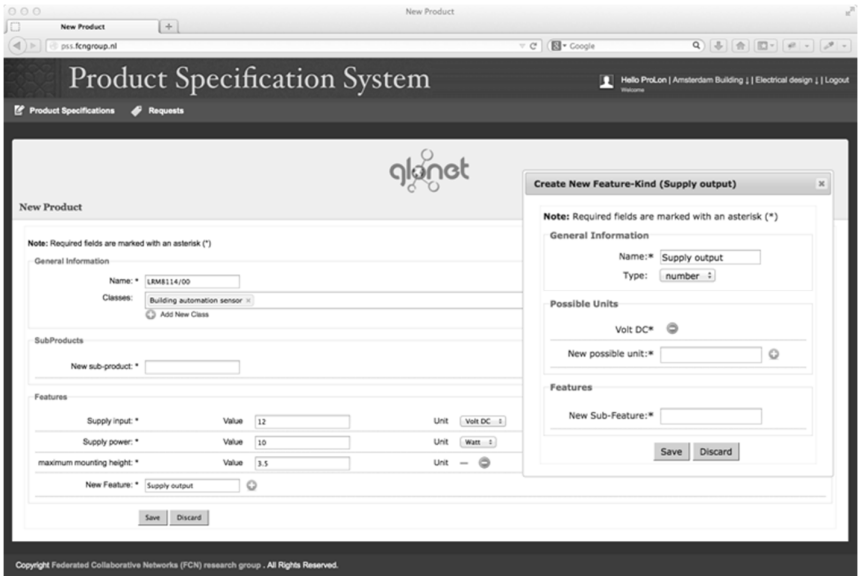

Figure 4 – Add New Specifications Window

4.2 Capturing product specification perspectives using classification

When specifying a sub-product related to a complex product, due to the multi-disciplinary nature of complex-products, it is important to enable the user by providing different perspectives of that sub-product, based on their related features. This can be supported through definition of classes in product specification. Furthermore, classes guide the users to provide proper feature information related to sub-products. For example one can define feature-kinds to be obligatory for a given class, so that if a user identifies a product as belonging to a certain class of products, then the user is warned to also provide features for its obligatory feature-kinds.

4.3 Supporting sub-product re-specification

When dealing with complex products, the user may wish to slightly re-specify a sub-product for its own design, or customize an existing sub-product specification in order to enhance, extend, revise, and finally perhaps assign it for restricted sharing. Several of the above needed functionality from the product specification space area already addressed in the paper, and represented in Figure 3. Two more functionality are required, as indicated in Figure 3 and described below.

Duplication action (shown with icon), which takes the user directly to a pre-filled "New Product" window. This simplifies the task of users, since in that window the specification information about the selected product is duplicated, which can be further modified/edited by the user to define a new but similar specification.

View action (shown with icon ), which takes the user to the view window of the product specification.

4.4 Complex product/sub-product Launch Request

After the designer has specified a product, its specification should be used for planning a VO that can configure and establish its realization. Figure 2 illustrates this functionality as the request for sub-product/complex product launch. This request can be issued by a sub-product designer, and must result in packaging of the product specification, and sending it through the cloud to the system that supports the VO formation for the sub-product.

The product specification space needs to support this functionality to enable the user with requesting the initialization/launching of the process that can realize the targeted specification. Thus, this request triggers the process of planning a goal-oriented VO.

4.5 Service specification and registration

Each business service (BS) is materialized through some business sub-processes [14]. These sub-processes represent *how* the services would be performed. The actions involved in the business service delivery can either be materialized automatically through some software (e.g. web services), manually through several human tasks, or even through a combination of these two kinds of activities. The automatic solutions are usually called software services, and the manual solutions are referred to as manual tasks. In order to develop a unified ICT-based business service specification environment, also for representing manual tasks we consider a simple software service that only indicates the start and end points of the corresponding task.

We consider four characterizing aspects of business services as being required to be provided during the service specification stage. These four aspects are required to improve functionalities supporting service interoperability, namely to support service discovery and service composition. These four aspects of the proposed service specification are described below, while the formalisms and standards that can be applied for representation of each aspect are also introduced. We have also adopted one specific notation for representation of each aspect, as also addressed below.

- Syntax: Typically, syntactic properties of a service are represented by XML-based standards and languages, such as the web service description language (WSDL) and Simple Object Access Protocol (SOAP) [15]. Some examples of syntactic aspects of the BS specification include: service name, the name of operations contained in the service, as well as their needed arguments. WSDL is selected as the notation for syntax specification in our development.

- Semantics: Conceptual properties of services, here referred to as semantics, are typically defined with ontology, as an explicit specification of a conceptualization of the knowledge about the service. The service ontology definition encompasses a group of vocabularies that specify semantic attributes of services (e.g. goals and category) and their inter-relationships, which together present a meaningful concept about the service [16]. In fact, the semantics description of BSs would enrich the lack of information about the services, which cannot be specified by syntactical descriptions, including: goals, context, pre-conditions and post-conditions of the BS. Here, OWL-S [17] is used for capturing service semantics within the proposed service specification. OWL-S

provides the rich description language needed for representing semantics related to services.

- Behavior: Besides semantics and syntactic description of the services, we also need to specify and formalize the externally observable behavior of each service, which shall represent the proper invocation order of its operations. These behavioral properties can be used later within the functions support service discovery and integration, for improving the accuracy of service matchmaking and facilitating the automation of integrated service execution [18], [19]. Furthermore, the behavioral aspects of the BS specification address its functionality, to the level that it can then be unambiguously implemented by software developers. We have proposed to formalize the behavior of the services in terms of Constraint Automata [20], within which every state of a Constraint Automaton (CA) represents an externally observable internal configuration of a service, and every transition represents the exchange of one or more messages by this service. In fact, a CA allows the user to capture the behavioral specification of a service by a finite number of states and some labelled transitions, as well as enabling software developers to follow the sequences of executed operations, in order to decide and implement the behaviors of the service. This behavioral specification comprises essential information for automated service invocation in case of stateful services [21]. Stateful services are defined where a client intends to keep either some data or some states during one invocation of the service, and then deploying those data and states during a subsequent invocation. In other words, the invocation of a stateful web service depends on its pervious invocations. To put it briefly, the formal specification of the stateful services' behavior provided by Constraint Automaton specifies the desired sequence for operations' invocation. The specification for stateless services consists of several single state CA, namely one Constraint Automaton for each operation in the service.

- Quality Criteria of Service (QCS): While the service discovery is usually done according to the functional properties of the BS specification (i.e. syntax, semantics and behavior of services), non-functional properties of services, i.e. Quality of Service (QoS) parameters also play an important role in customer's service selection. Therefore, we have also proposed to specify some QoS metrics as quality criteria of services to assist customers in service selection and to improve the accuracy and optimization in service matchmaking. The QoS values of services are usually claimed by service providers and ensured through a service level agreement (SLA) as a part of a contract between the service provider and the customers [22]. We have identified some quality criteria for assessment of offered services such as the execution duration, the maximum response time, and the service availability. The QCS agreements in SLAs are represented as promises among the involved partners in the VO. In [23] different states of such promises are introduced, including: conditional, unconditional, kept, not kept, withdrawn, released, and invalidated.

6 Concluding Remarks

This paper addresses the area of service enhanced complex product specification within the context of VBEs and goal-oriented VOs, which involve collaborations among

competing companies, the so called coopetition. We presented a set of functional and non-functional requirements for product and service specification within this context and in different phases of the complex products life cycle. Furthermore we have introduced and provided some design and implementation details for developing a coopetition space to support complex product specifications, while realizing its identified non-functional and functional requirements. The implementation of this system has been developed in Java programming language, using the Spring [24] and Hibernate [25] frameworks. Its database is built using the GloNet platform [10] and the MySQL [26] database management system. The general framework applied for development of the complex product specification space follows the layer-based MVC (Model–View–Controller) software design pattern [27]. This system is already beta tested by industrial partners within the GloNet project. Some more details over its design and implementation are presented in [28].

Acknowledgments. This work was funded in part by the European Commission through the GloNet project (FP7 programme). The authors also thank the contributions from their partners in the project.

References

1. Li, W. D., Fuh, J. Y., & Wong, Y. S. (2004). An Internet-enabled integrated system for co-design and concurrent engineering. *Computers in Industry*, *55*(1), 87-103.
2. Li, W. D., Lu, W. F., Fuh, J. Y., & Wong, Y. S. (2005). Collaborative computer-aided design—research and development status. *Computer-Aided Design*, *37*(9), 931-940.
3. Hertog, P. D. (2000). Knowledge-intensive business services as co-producers of innovation. International Journal of Innovation Management, 4(04), 491-528.
4. Bitner, M. J., & Brown, S. W. (2006). The evolution and discovery of services science in business schools. Communications of the ACM, 49(7), 73-78.
5. Afsarmanesh, H., Sargolzaei, M., & Shadi, M. (2014). Semi-automated Software Service Integration in Virtual Organizations. International Journal of Enterprise Information Systems "Service-based Interoperability and Collaboration for Enterprise Networks", (in print).
6. Camarinha-Matos, L. M., Afsarmanesh, H., & Koelmel, B. (2011). Collaborative networks in support of service-enhanced products. In Adaptation and Value Creating Collaborative Networks (pp. 95-104). Springer Berlin Heidelberg.
7. Afsarmanesh, H., & Thamburaj, V. (2012). ICT requirements analysis for enterprise networks supporting solar power plants. In Collaborative Networks in the Internet of Services (pp. 149-157). Springer Berlin Heidelberg.
8. Abran, A., Bourque, P., Dupuis, R., & Moore, J. W. (2001). Guide to the software engineering body of knowledge-SWEBOK. IEEE Press.
9. Kaufman, C., Perlman, R., & Speciner, M. (2002). Network security: private communication in a public world. Prentice Hall Press.
10. Surajbali, B., Bauer, M., Bär, H., & Alexakis, S. (2013). A Cloud–Based Approach for Collaborative Networks Supporting Serviced-Enhanced Products. In Collaborative Systems for Reindustrialization (pp. 61-70). Springer Berlin Heidelberg.
11. Lubbers, P., Albers, B., Salim, F., & Pye, T. (2011). Pro HTML5 programming (pp. 107-133). New York, NY, USA:: Apress.

12. McCormick, E., & De Volder, K. (2004, October). JQuery: finding your way through tangled code. In Companion to the 19th annual ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications (pp. 9-10). ACM.
13. Hvam, L., & Ladeby, K. (2007). An approach for the development of visual configuration systems. Computers & Industrial Engineering, 53(3), 401-419.
14. Camarinha-Matos, L. M., Afsarmanesh, H., Oliveira, A. I., & Ferrada, F. (2013). Collaborative Business Services Provision. In Proceedings of ICEIS'13–15th International Conference on Enterprise Information Systems (Vol. 2, pp. 382-392).
15. Curbera, F., Duftler, M., Khalaf, R., Nagy, W., Mukhi, N., and Weerawarana, S., 2002. "Unraveling the Web services web: an introduction to SOAP, WSDL, and UDDI". Internet Computing 6(2):86-93.
16. Afsarmanesh, H. & Ermilova, E. (2010). The management of ontologies in the VO Breeding Environments domain. International Journal of Services and Operations Management, 6(3), 257-292.
17. Lara, R., Roman, D., Polleres, A., and Fensel, D. 2004. "A conceptual comparison of WSMO and OWL-S". In Proceedings of the European Conference on Web Services, volume 3250 of Lecture Notes in Computer Science, September 2004, Erfurt, Germany.
18. Sargolzaei, M., Santini, F., Arbab, F., & Afsarmanesh, H. (2013). A tool for behaviour-based discovery of approximately matching web services. In Software Engineering and Formal Methods (SEFM 2013), LNCS 8137 (pp. 152-166). Springer Berlin Heidelberg.
19. Afsarmanesh, H., Sargolzaei, M., & Shadi, M. (2012). A framework for automated service composition in collaborative networks. In Collaborative Networks in the Internet of Services (pp. 63-73). Springer Berlin Heidelberg.
20. Baier, C., Sirjani, M., Arbab, F., and Rutten, J., 2006."Modeling Component Connectors in Reo by Constraint Automata". Science of Computer Programming, Elsevier 61(2): 75-113.
21. Jongmans, S. S. T., Santini, F., Sargolzaei, M., Arbab, F., & Afsarmanesh, H. (2012). Automatic code generation for the orchestration of web services with Reo. In Service-Oriented and Cloud Computing (pp. 1-16). Springer Berlin Heidelberg.
22. Dan, A., Davis, D., Kearney, R., Keller, A., King, R., Kuebler, D., Ludwig, H., Polan, M., Spreitzer, M., Youssef, A.: Web services on demand: Wsla-driven automated management. IBM systems journal 43(1), 136–158 (2004).
23. Shadi, M. and Afsarmanesh, H., 2013. "Agent Behavior Monitoring in Virtual Organizations". In Proceedings of the 22th IEEE International Workshops on Enabling Technologies (WETICE 2013), 17-20 June 2013, Hammamet, Tunisia.
24. Johnson, R., Hoeller, J., Arendsen, A., & Thomas, R. (2009). Professional Java Development with the Spring Framework. John Wiley & Sons.
25. Bauer, C., & King, G. (2006). Java Persistance with Hibernate. Dreamtech Press.
26. Kofler, M. (2001). What Is MySQL? (pp. 3-19). Apress.
27. Reenskaug, T. (1979). Models-views-controllers. Technical note, Xerox PARC, 32, 55.
28. Afsarmanesh, H., & Shafahi, M. (2013). Specification and Configuration of Customized Complex Products. In Collaborative Systems for Reindustrialization (pp. 81-90). Springer Berlin Heidelberg.