D4.1 Design report on approach and mechanism for effective customized complex product specification
Afsarmanesh, H.; Shafahi, M.; Ünal-Karakas, Ö.; Sargolzaei, M.

Citation for published version (APA):

General rights
It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations
If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: http://uba.uva.nl/en/contact, or a letter to: Library of the University of Amsterdam, Secretariat, Singel 425, 1012 WP Amsterdam, The Netherlands. You will be contacted as soon as possible.
Glocal enterprise network
focusing on customer-centric collaboration

D4.1
Design report on approach and mechanism for
effective customized complex product specification

Edited by

UvA
February 2013

Project funded by the European Commission under the
ICT – Factories of the Future Programme

Contract Start Date: 1 Sep 2011  Duration: 36 months
Deliverable summary

This deliverable addresses the design of an approach and a set of mechanisms for effective customized specification of complex products, and thus it reports on the findings of the Task 4.1 in workpackage 4.

The deliverable first identifies the generic entities and concepts needed for specification of the three types of complex products addressed in GloNet, namely: the PV power plants, the intelligent buildings, and the future incubators. Then a set of generic meta-data is specified for representing data related to detailed configuration and ordering of these complex products. A class diagram is presented corresponding to this specification, within which a set of classes, their attributes, as well as the methods defined on the classes are specified. Both the functional and non-functional requirements of the aimed complex products are identified and analysed. Through analysing these requirements, a set of 21 use cases are designed. Furthermore, for each use case, its corresponding UML use case diagram and some textual description are provided. Then in the scope of the database design, a relational database schema is defined for managing all data in this system. Finally, a set of data manipulation services for the database system are designed, which will be implemented in the next task T4.2. As such, a number of representative abstractions of user interfaces are designed representing the base data manipulation functionalities and interactions of the user with the system.

In the next step of WP4, during the task T4.2, the prototype of the database system addressed in T4.1, as well as its planned services will be implemented. This development aims at supporting the effective complex product specification, during its gradual process, and implements the design described in this deliverable. The data manipulation services running on the database will be cloud-based software services that will be developed on top of the CAS platform.
<table>
<thead>
<tr>
<th>TABLE OF CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROJECT-RELATED SUMMARY ................................................................................................................. 4</td>
</tr>
<tr>
<td>1. INTRODUCTION .......................................................................................................................................... 5</td>
</tr>
<tr>
<td>2. MOTIVATION AND APPROACH .................................................................................................................. 6</td>
</tr>
<tr>
<td>3. Advanced object-oriented modeling of main entities and concepts .......................................................... 9</td>
</tr>
<tr>
<td>3.1 Complex product and its atomic and composite sub-products .................................................................. 9</td>
</tr>
<tr>
<td>3.2 Feature-kind and features .................................................................................................................... 10</td>
</tr>
<tr>
<td>3.2.1 Atomic feature-kinds ...................................................................................................................... 10</td>
</tr>
<tr>
<td>3.2.2 Composite feature-kinds .................................................................................................................. 10</td>
</tr>
<tr>
<td>3.3 Offers ................................................................................................................................................... 10</td>
</tr>
<tr>
<td>Figure 5. The augmented process of purchasing a sub-product .................................................................. 11</td>
</tr>
<tr>
<td>3.4 Object-classes ...................................................................................................................................... 12</td>
</tr>
<tr>
<td>3.4.1 Object-classes for sub-products .................................................................................................... 13</td>
</tr>
<tr>
<td>3.4.2 Object-classes for offers .................................................................................................................. 13</td>
</tr>
<tr>
<td>3.6 Relationships among the entities and concepts .................................................................................... 14</td>
</tr>
<tr>
<td>4. Analysis of Requirements for product specification system ..................................................................... 15</td>
</tr>
<tr>
<td>4.1 Non-Functional Requirements ........................................................................................................... 15</td>
</tr>
<tr>
<td>4.1.1 Security and authentication ........................................................................................................... 16</td>
</tr>
<tr>
<td>4.1.2 Scalability ....................................................................................................................................... 16</td>
</tr>
<tr>
<td>4.1.3 Portability ........................................................................................................................................ 16</td>
</tr>
<tr>
<td>4.2 Functional Requirements ..................................................................................................................... 17</td>
</tr>
<tr>
<td>4.2.1 Actors: Stakeholders in the system .................................................................................................. 17</td>
</tr>
<tr>
<td>4.2.2 Use Cases ...................................................................................................................................... 18</td>
</tr>
<tr>
<td>4.2.3 Use Case 1 - Managing offers ......................................................................................................... 19</td>
</tr>
<tr>
<td>4.2.4 Use Case 2 - Adding a new Offer (Offering a Product) ................................................................... 21</td>
</tr>
<tr>
<td>4.2.5 Use Case 3 - Editing an Offer ......................................................................................................... 22</td>
</tr>
<tr>
<td>4.2.6 Use Case 4 – Changing the status of an Offer ............................................................................... 22</td>
</tr>
<tr>
<td>4.2.7 Use Case 5 - Viewing an Offer ...................................................................................................... 23</td>
</tr>
<tr>
<td>4.2.8 Use Case 6 - Managing Products .................................................................................................. 23</td>
</tr>
<tr>
<td>4.2.9 Use Case 7 - Adding a new Product ............................................................................................... 24</td>
</tr>
<tr>
<td>4.2.10 Use Case 8 – Duplicating and Editing a Product ........................................................................... 25</td>
</tr>
<tr>
<td>4.2.11 Use Case 9 - Viewing a Product .................................................................................................. 25</td>
</tr>
<tr>
<td>4.2.12 Use Case 10 - Managing Object-classes ...................................................................................... 26</td>
</tr>
<tr>
<td>4.2.13 Use Case 11 - Adding a new Object-class ................................................................................... 27</td>
</tr>
<tr>
<td>4.2.14 Use Case 12 – Duplicating and editing an existing Object-class ..................................................... 27</td>
</tr>
<tr>
<td>4.2.15 Use Case 13 - Viewing an Object-class ....................................................................................... 28</td>
</tr>
<tr>
<td>4.2.16 Use Case 14 - Managing Feature-kind ......................................................................................... 28</td>
</tr>
<tr>
<td>4.2.17 Use Case 15 - Adding a new Feature-kind .................................................................................... 29</td>
</tr>
<tr>
<td>4.2.18 Use Case 16 – Duplicating and Editing a Feature-kind .................................................................. 29</td>
</tr>
<tr>
<td>4.2.19 Use Case 17 - Viewing a Feature-kind ......................................................................................... 30</td>
</tr>
<tr>
<td>4.2.20 Use Case 19 - Adding a new Scale ............................................................................................... 31</td>
</tr>
<tr>
<td>4.2.21 Use Case 20 – Duplicating and Editing a Scale ............................................................................. 32</td>
</tr>
<tr>
<td>4.2.22 Use Case 21 - Viewing a Scale .................................................................................................... 32</td>
</tr>
<tr>
<td>5. Database Design ................................................................................................................................... 33</td>
</tr>
<tr>
<td>5.1 Relational database schema design ..................................................................................................... 33</td>
</tr>
<tr>
<td>6. User interaction and interfaces design ................................................................................................... 34</td>
</tr>
<tr>
<td>6.1 User interface functionality design .................................................................................................... 34</td>
</tr>
<tr>
<td>6.1.1 Product .......................................................................................................................................... 34</td>
</tr>
<tr>
<td>6.1.2 Offer ............................................................................................................................................. 35</td>
</tr>
<tr>
<td>6.1.3 Feature-kind .................................................................................................................................. 36</td>
</tr>
<tr>
<td>6.1.4 Object-class .................................................................................................................................. 36</td>
</tr>
<tr>
<td>6.1.5 Scale ............................................................................................................................................. 37</td>
</tr>
<tr>
<td>7. CONCLUSIONS ...................................................................................................................................... 38</td>
</tr>
<tr>
<td>8. REFERENCES ....................................................................................................................................... 39</td>
</tr>
<tr>
<td>ANNEX - A ............................................................................................................................................... 40</td>
</tr>
</tbody>
</table>
This deliverable is the first outcome of WP4. It plays an important role for the next deliverables of this workpackage, as well as for interconnecting WP4 with other workpackages. Being produced as the result of Task 4.1, as a **first main step** in complex product configuration, this deliverable corresponds to the design document for the effective customized product specification. Findings reported in deliverables: D1.1 (“Detailed requirements for GloNet use case and domain glossary”), D1.2 (“Specification of business scenarios”), and D2.1 (“Required information/knowledge provision services specification”) constitute the main inputs for this deliverable.

As shown in the following figure, the system designed in this deliverable will be implemented in the second task of WP4, namely the task T4.2.

Later on, in the task T4.3 addressing the **second main step** in complex product configuration, the design described in D4.1 will be further expanded with the introduction of intelligent tools needed for automated recommendation of the complex product configuration. The approach that will be introduced in T4.3, will be based on case-based reasoning and learning, in order to automate the predication of needed sub-products and recommendation of sub-products that can best fulfill the complex product requirements.

As such, deliverable D4.1 provides the base design of the main skeleton needed for developing a complex product specification / configuration system, as well as the required underlying database model and data manipulation services for this system. Therefore, establishing a good design in this deliverable is imperative to the success of the other three deliverables of WP4.

Furthermore, the results generated by WP4 system constitutes one input for VO creation process in WP5.
1. INTRODUCTION

Both the configuration process and the specification of products are addressed in computer science and systems engineering research. One commonly used definition for the configuration process is provided in [1]:

“Given: (i) a fixed, pre-defined set of components, where a component is described by a set of properties, ports for connecting it to other components, constraints at each port that describe the components that can be connected at that port, and other structural constraints (ii) some description of the desired configuration; and (iii) possibly some criteria for making optimal selections.

Build: One or more configurations that satisfy all the requirements, where a configuration is a set of components and a description of the connections between the components in the set, or detect inconsistencies in the requirements.”

Product configuration process has also been widely studied in different application domains, and especially in manufacturing [2]. When considering the increasing rate of competition among the companies in the market, as well as the variety and number of involved stakeholders, the role of presenting the customer with a configured product that best meets his/her requirements and at the lowest possible cost, has become increasingly significant.

As the first main step in this process (as addressed in this document), in order to make the configuration process less complex for the involved stakeholders, there is a need to provide the user with effective and iterative mechanisms and tools for product specification. But the process of configuring and ordering complex products, such as those addressed in the GloNet Project (i.e. PV power plants, intelligent buildings, and future incubators), raises a number of other challenges. These challenges are due to many reasons, such as the following:

- A complex product is made of multiple sub-products at different levels of granularity
- A complex product is fully customized per-order
- A complex product must be designed, monitored, and get approved by multiple stakeholders
- A complex product is gradually configured
- Related to every sub-product, there might typically be several offers, e.g. either directly from their manufacturers/producers or from other nodes (here called suppliers/sellers) that provide such sub-products to the market
- etc.

In order to help the customer in co-configuring and ordering of the complex product, as well as its related sub-products and business services, in this document we aim to design a generic reusable approach and mechanism for effective product specification, so that different involved stakeholders can effectively customize the complex products based on both their preferences and mandatory criteria. Toward achieving this goal, after presenting the main motivations in section 2, the definitions of a number of important entities and concepts are provided in Section 3, in order to provide a clear understanding of the proposed system. A class diagram, showing the relationships among the defined concepts is presented in the same section. Then, the results of our analysis of the functional and non-functional requirements related to this system are provided in Section 4. A generic relational database schema, which aims to represent the data related to configuration of different types of complex products in GloNet, is presented in Section 5. Finally, the results of the user interface design process are provided in Section 6. The designed user interface presents the data manipulation services that will be implemented in WP4, and gives a feeling for the looks of the implemented aspects from the usage point of view.
2. MOTIVATION AND APPROACH

The GloNet project is targeted at supporting complex products, and specially the PV power plants as the guiding use case, but also providing generic solutions that can be applied to the case of intelligent buildings, and future incubators. Every such complex product is composed of a set of different sub-products (e.g. equipment) and sub-systems (e.g. software-based services) provided by different suppliers. Furthermore, complex products require high degree of customization. Customers need products customized to their unique requirements and preferences. They need assistance to select the best combination of sub-products that meet their requirements as well as the best fit business services to support their product. Typically, business services are partly provided manually (e.g. the cleaning of vegetation in solar panels), which are referred to as manual services in this text, and partly augmented by some software systems (e.g. calibration of sensors), which are here referred to as software-based services. Furthermore, at different stages of complex product’s life cycle, different needs arise for ordering needed sub-products. For instance, during the complex product’s creation stage, large number of inter-related sub-products must be considered and addressed, e.g. equipment, business services, and software-based services that are involved in the configuration of the complex product. This in turn requires careful selection of the best fit sub-products to compose the offer that support the stated requirements. Also, during the operation and maintenance of the complex products, more specific sub-products are needed to be planned and considered. Thus, it is imperative to provide the customers with tools to specify the details of required sub-products for their complex product, including their required business services. To achieve this goal, a main step involves capturing customer requirements for the complex product. Clear specification of customer requirements, typically an incremental process, enables the project developer to offer the most-fit corresponding complex product to meet customer’s requirements. Therefore, the main subject of this document is targeting the effective design of a complex product specification system to support this process.

The important task of providing customers with some intelligent tools that can learn from the past orders, in order to suggest a number of sub-product alternatives will be achieved in the later stages of the GloNet project, and as a follow up to the current tasks addressed in this deliverable.

For the purpose of designing the product specification system, while using as the base the results of the requirements analysis task preformed for solar power plants [3], we first identify the generic set of entities and concepts common to the three different types of complex products in GloNet. Second, mainly based on the initial analysis of the requirements addressed in [3], as well as on the designed business scenarios addressed in [4], and the introduced information/knowledge provision services addressed in [5], which are resulted in the previous steps of the GloNet Project, we extract and analyze here the functional and non-functional requirements for the proposed product specification system. Third, we specify a generic data model for building the database, to capture the data related to different orders of sub-products and sub-systems constituting the complex-products. Since this model is generic, it can be used for different complex-products. Fourth, at the final stage, we design this complete product specification system, in terms of its database, user interface, and architectural design. These steps are shown in Figure 1.

The proposed product specification system is planned to be developed on top of the GloNet cloud-based platform [6]. As such, stakeholders will be supported by cloud-based services provided through user-friendly interfaces, to effectively specify their products, customized to their needs. The platform will provide a number of secure information/knowledge access services as the base for the aimed functionalities of the product specification system. Figure 2 shows the high level architecture offered through the proposed product specification system, of which the design is addressed in this document. In the solar power plant case for instance the
project developer and/or the EPC, works closely with the customer to configure the needed complex product, and in order to offer the product that satisfies all the needs and preferences of the customer.

Figure 1. Steps followed in design of the product specification system

Complex product specification is not a one-step process; rather it is typically achieved gradually and in a number of sessions. For this reason, two main basic operations are designed to support this aspect, which includes: 1) operation used to introduce/add a new product specification, 2) operation used to edit and extend the specified product data previously entered. While the first operation will be used only once for definition of each new sub-product, sub-system etc., the second operation will be used repetitively, until the completion of the complex product specification. After the product specification is completed, the specified data needs to be viewed and accepted (or rejected) by the Customer, assisted by a Consultant and/or project developer. In case of rejection, a part of the specification may need to be further refined and/or redefined at a later stage. In addition to supporting the specification of the product design, besides the two above mentioned operations, a number of other auxiliary functionalities are also needed to be provided through the product specification system, details of which are provided in the use case descriptions of Section 4.

Figure 2. High level architecture of the Complex Product Specification System
The important features aimed by the design of the product specification system can be summarized into the following:

- **Generality and reusability** – The designed meta-data (schema) for the database system, needed for storing product specification details, must be generic. The aimed generality/reusability is in the sense that they should be able to model any entity and concept belonging to any complex product, and not dedicated to supporting only one or two specific complex products.

- **Usage simplicity** – While the designed functionality for the product specification system can be complex, the designed interfaces for using these functionalities must hide the system complexity and provide easy to use operations.
3. ADVANCED OBJECT-ORIENTED MODELING OF MAIN ENTITIES AND CONCEPTS

To properly design the complex product specification system, we first have to identify the main type/nature of the entities and concepts that are specific to complex product specification. To identify such type or nature for entities and concepts, we start by applying the results of detailed requirements analysis stage of solar power plants addressed in [3]. Furthermore, we extend this base in discussion with other GloNet consortium partners, experienced in the domains of: solar power plants, intelligent buildings, and future incubators. In the following sub-sections, we define a set of generic entities and concepts, and apply advanced object-oriented approaches to model them. The defined models are generic in the sense that they can support the full specification of all possible different types of complex products which may ever need to be defined in GloNet. Furthermore, the approach is generic. This means that the designed system can support the complete specification of a complex product, with all its extensive offer generation (what is possibly needed to be developed by a project developer for the approval of the customer or the consultant of the customer) during the creation stage of the complex products. At the same time, the same designed system can also support the simple specification of new equipment that might be needed during the maintenance stage of the solar plant (what is possibly directly negotiated between the customer or the EPC representing it based on an offer for a specific product directly from the supplier or manufacturer). Please note that first section 3.6 and figure 9 represent a summary of the modeled objects and their inter-relationships. But later on in Section 5, a detailed definition of these modeled objects and the complete schema design of the database supporting the product specification system is provided.

3.1 Complex products and their atomic and composite sub-products

A complex product is composite and consists of a set of sub-products. A sub-product may represent a specific piece of equipment, a business service or a software-based service, that is produced and/or provided in the industry and is used for generating the complex product. Sub-products can themselves be composite, in the sense that they may consist of other sub-products. For example an intelligent building is a complex product, including large number of equipment sub-products, for instance an intelligent thermostat, while the intelligent thermostat itself consists of other equipment including a thermometer, etc.

In this document, we only refer to complex product when we are talking about one of the 3 kinds of complex products in the GloNet Project, namely the PV power plants, intelligent buildings, or future incubators. Sub-products then refer to all other constituents of the complex product, being themselves atomic or composite, and please note that we classify sub-products under the name of product in our data model.

Sub-products have a set of known characteristics, which we model with feature-kind in this document, representing the “kind/type” for their characteristics/features. For example, a Pyranometer, the sub-product shown in Figure 3, can be characterized by its following feature-kinds, each are also exemplified below by one specific feature instance for this sub-product:

1. Manufacturer (such as iPLON)
2. Minimum light sensitivity (such as 7 µV/W/m2)
3. Maximum light sensitivity (such as 14 µV/W/m2)

Furthermore, the aggregation of a set of products identifies/forms an object-class for those products. Similarly the aggregation of a set of feature-kinds identifies/forms an object-class for those feature-kinds. As such, each real physical sub-product may be an instance of several object-classes, where each may characterize different set of its feature-kinds.
3.2 Feature-kind and features

A feature-kind is a characteristic of a sub-product, and may have multiple scales. One specific value together with the scale for a feature-kind of a sub-product constitutes a tuple that we call a feature of the sub-product. The combination of all the features of a sub-product makes that sub-product unique. For example the atomic sub-product lamp has the feature-kinds of: input voltage, amount of energy it consumes, amount of light it should produce, and the color of its light beams. But the definition of lamp is not unique until we indicate that it has an input voltage of 220 volts, consumes 100 watts of energy and provides 800 lumens of blue light. Therefore, the latter definition of lamp fully represents one real physical lamp (traditionally called an instantiation of the type lamp), while the former definition of lamp represents only the meta-data for type lamp in traditional database descriptions. Feature-kinds themselves could be also atomic or composite, as explained further below.

3.2.1 Atomic feature-kinds

An atomic feature-kind is a feature-kind that has a specific name and type, with one or multiple scales defined for it. For example the “width” is an atomic feature-kind with the type float, and the two scales of centimeters and millimeters. Furthermore, one specific feature e.g., the “width of a power cord” has the single scale millimeter.

3.2.2 Composite feature-kinds

A composite feature-kind constitutes a set (collection) of either atomic feature-kinds or other composite feature-kinds. For example, the feature-kind “location” could have two sub-feature-kinds, namely “latitude” and “longitude”. Each composite feature-kind may therefore have a set of possible scales.

3.3 Offers

An offer is either an agreement discussed between two stakeholders, or a special deal on a sub-product suggested by one stakeholder to public (i.e. all potential buyers). In the former case, during the creation phase of the complex product, agreements on the details of sub-products etc. are gradually reached between the project developer/EPC and the customer (or its consultant), constituting the offer under negotiation between these two entities, related to the configuration of the complex product.

Another kind of offer, the latter case, is for instance the specification of a special deal offered by a supplier/seller for a new sub-product. For example, an offer may include a 1MWp cabinet, to be delivered in Paris, on March 15th, with the price of 10000 Euros. Multiple offers could also be defined.
for one possible sub-product. For example, a sub-product could be offered by a seller with the price of 10 euros if it is delivered in Amsterdam, but 15 euros if delivered in Dusseldorf. Following gives a set of differences between the “sub-product” entity and the “offer” entity, when the offer consists of specification of a special deal. We therefore model these two entities differently in the GloNet complex product specification system:

- Supplier/seller of a sub-product may create a sub-product specification offer, while the sub-product itself is produced by its equipment manufacturer and/or a service provider.
- Offer specification contains information about the supplier/seller, while the sub-product specification contains information about the manufacturer/provider.
- Offer specification contains the sale conditions, geographical location for delivery, pricing, payment, etc. (as indicated in the example above), while the sub-product specification indicates the production aspects and conditions.
- ...

Figure 4 shows the general process of where a special deal offer appears in the whole procedure of purchasing a sub-product.

![Figure 4. The process of purchasing one sub-product](image)

In our model, we classify both the manufacturers of equipment and the providers of services as the producing/providing organizations. Based on Figure 4, the steps in the simple case of purchasing a sub-product can be augmented in our system with the following steps:

1- A sub-product is specified by its producer/provider
2- One or multiple offers may be specified (made) by different sellers, for each sub-product
3- The customer may select one of the offers and pays for it, which is then delivered to her/him.

Figure 5 demonstrates the augmented process of making and approving the purchase of the above sub-product based on its offer(s).

![Figure 5. The augmented process of purchasing a sub-product](image)

Although the process of purchasing one sub-product is usually simple, the process of purchasing a complex-product is much more complicated. It involves configuring the complex product involving both the configuration of its sub-products. Finally it involves configuring a final offer for the complex-product, so that the customer can approve or reject it. In all the sub-steps mentioned for the process of purchasing a complex-product the customer should be involved, so that the final result (the final complex-product) meets his/her requirements and that the final offer can be accepted by the customer.
Figure 6 demonstrates the process of making an offer with all the details that need to be planned for a complex product required by the customer.

Please note that every accept or reject stated on a sub-product offer (e.g. planned by the project developer) also extends the profile which GloNet keeps for characterizing the customer, what will be used in the next steps of this work, to dynamically narrow down the product alternatives that will be offered to each customer.

![Figure 6. The process of offering a complex-product](image)

### 3.4 Object-classes

As addressed before, the feature-kinds of a sub-product together represent a specific classification for that sub-product in the system. This is also true for the feature-kinds of the offers in the system. We introduce **object-class** as a basic class defined to model the generic categorization of all (sub-) products, and offers, where each object-class (for a sub-product or offer) has a specific set of feature-kinds.

Furthermore, we can enforce a set of obligatory feature-kinds to sub-products and/or offers. For example, definition of an object-class “Electrical” (a class representing certain kind of sub-products and/or offers) can simplify the search for the sub-product in the category of “Electrical”, while at the same time it can enforce the definition of an obligatory feature-kind, such as “voltage” to this class. In other words, any sub-product class that has the object-class “Electrical”, must have the “voltage” feature-kind and therefore any instance of that sub-product class must have a feature (e.g. value) for its voltage.

Figure 7 shows four different example object-classes that are defined for sub-product classes in in PV power plants, and represents a set of features-kinds that are specified for each of these object-classes, e.g. sensor object-class has a feature-kind accuracy.
Furthermore, object-classes can be used to identify and filter what feature-kinds are relevant to which stakeholder. For example, the feature-kind “life span” might be interesting for an insurance company, but it might not be interesting for other stakeholders e.g. government regulators, as demonstrated in Figure 8. These aspects will be supported through the definition of database views.

3.4.1 Object-classes for sub-products

In addition to what is mentioned above, introduction of object-classes in our database model provides the following specific benefits for sub-product definitions:

1. Flexibility to categorize the same (real physical) sub-product as different kinds of sub-products (e.g. electrical, optical, etc.)
2. Guaranteeing that the user will provide the required (obligatory) input for each sub-product
3. Ensuring efficient and easy access to product information for different stakeholders

3.4.2 Object-classes for offers

Object-classes are also used to categorize offers related to sub-products. They can also enforce a set of obligatory and optional feature-kinds for the offer. For example, the object-class “Location Based” can enforce the obligatory feature-kind “Geo-coordinates”. In other words, any offer that has the
object-class “Location Based”, must have the “Geo-coordinate” feature-kind defined for it, and therefore any instance of this sub-product must have a feature (e.g. value) for its Geo-coordinate, which could later be used to identify offers that are being made for a specific region.

In summary, object-classes provide the following advantages for offers:

1. Flexibility to define different kinds of offers (being Geo-based etc.)
2. Having the option to filter provided offers by the requested needs of the customer
3. Ensuring efficient and easy access to offer information for different stakeholders

3.6 Relationships among the entities and concepts

In order to clearly specify and analyse the requirements of our complex product specification system we should formalize the object-oriented model described in previous subsections. That means we should model all needed entities and concepts involved in the system and define their interrelationships. Figure 9 provides a summary class diagram (representing only the class names and their inter-relationships), which are defined as the meta-data for our model. These classes model all the entities and concepts in the product specification system, while satisfying the requirement of being generic and reusable. It is important to note that in this diagram the class “Object-class” models all sub-products and offers, while it aggregates a set of feature-kinds related to the definition of each sub-product or offer. Furthermore, the “Product” class represents any sub-product of a complex product, as well as ultimately the complex product itself.

[Diagram of class relationships]

Figure 9. Summary class diagram of modeled object in product specification system

Annex – A of this document provides the complete UML class diagram for which a summary is presented in Figure 9.
4. ANALYSIS OF REQUIREMENTS FOR PRODUCT SPECIFICATION SYSTEM

System requirements address why a system is needed, what are the functions it must accomplish, and how the system is to be constructed and implemented and what conditions must be satisfied by the system [7]. System requirements analysis is one of the steps of the requirements engineering process, and is imperative for well establishment of the needs of a new system to be developed, and leads to a well-designed and well-implemented system. Therefore, this is a critical step for the success of our software system [8]. Requirements analysis also includes activities for detecting and resolving conflicts between the requirements, classifying them, discovering the boundaries of the system, how it must interact with its environment, and elaborating on the system requirements in order to derive software requirements [9].

The two main types of requirements to be considered are the non-functional and the functional requirements. Non-functional requirements are those that are not directly related to the functions of the system, such as the security, scalability, availability, etc. These are primarily related to software provision constraints and quality requirements. Functional requirements on the other hand, define the necessary functions expected from the system and they describe the input and outputs of the system.

One of the mostly used tools for documenting the functional requirements of a system are the use cases. Use cases show how the system interacts with the user or another system. These interactions are represented as a Use Case Diagram in UML. Furthermore, users or other systems interacting with the system are addressed as “actors”.

In order to systematically design the product specification system, we have used both the requirements elicited at the initial phase of the GloNet Project [6], as well as the results of our further discussions with the industry partners of GloNet consortium representing the three complex product types. We can therefore properly analyze the requirements for this system.

In this section, the results of this analysis are presented in terms of both non-functional and functional requirements. As for documenting the functional requirements, we use UML use case diagrams and use case descriptions. Since there are no standard notation for providing the formal descriptive definition of the use cases, in Section 4.2 we augment the use case diagrams with a template which includes the following extra elements, which through their (textual) descriptions enhance the definition of the use cases:

- **Use Case Name**: Short name for the use case
- **Description**: Short textual description of the use case
- **Actor**: Main user(s) involved in this use case
- **Pre-condition**: The condition that needs to be fulfilled in order for the use case to start
- **Normal Flow**: For describing the steps involved in the normal flow of the use case, i.e. when there are no errors and/or exceptions
- **Alternative Flow**: For describing the steps followed when an error or exception occurs in the system during the normal flow

### 4.1 Non-Functional Requirements

Many non-functional requirements can be mentioned for the product specification system. Among them, we have selected the most important ones and will shortly describe them in the following subsections.
4.1.1 Security and authentication

In a system that is dealing with industrial profit we should be careful about threats both from the outside world and from the inside of the system (via the competitors). Namely, security and authentication must be preserved between different users as well as for the software modules running inside the system, to prevent any malicious access and to prohibit endangering the privacy of involved stakeholders.

4.1.1.1 Securing from the external world

It is obvious that the parties from the outside world might be interested in gaining access to the system and the information stored in the database of the system or in preventing the proper operation of the system. This is due to the fact that doing so might provide them with financial benefits, like competing with stakeholders involved in a VO. In order to prevent this, effective authentication mechanisms should be applied before allowing access rights to a user. Other than that, the infrastructure should also be resistant to attacks; such as the refusing and halting software attacks and preventing unauthorized stakeholders from accessing the system. This requirement is met by using the GloNet platform [6] that is well built to resist attacks and support proper authentication.

4.1.1.2 Securing from other internal stakeholders

One of the most important requirements of a system that is dealing with multiple stakeholders is its secure and proper information sharing, since although these stakeholders cooperate to achieve a common specific goal, they can also be potential competitors on other goals. In other words, it is necessary to only share that part of the information between the stakeholders that is required for achieving their common goal. It is also important to prevent any kind of information leakage by means of malfunctioning software, both to the outside world and between the stakeholders. This requirement is achieved by providing proper authorization for the stakeholders and the information they share in the system. In order to meet this requirement, libraries and tools provided by the underlying GloNet platform [6] can be used.

4.1.2 Scalability

Since the system could potentially be supporting a VBE and dealing with hundreds of users, it is important that it could be easily extended. The fastest way of achieving this is building the system in such a way that it benefits from the cloud environment and allows allocation of resources on demand. Since the product specification system will be implemented on top of a cloud-based platform, namely the GloNet platform [6], this system will inherit the scalability feature of the platform.

4.1.3 Portability

Since the product specification system will be developed as a web-based application that will be written using the Java language, both the server and the client side of the system will be portable. Considering the client side, a user can easily access the system using any web browser that is nowadays available on almost all systems such as PCs, Laptops, and mobile phones. As for the server side, since the program will be written in Java, it can be executed on any platform that provides the Java virtual machine environment.
4.2 Functional Requirements

These requirements address the expected functions to be provided by the system, together with any specific constraints on these functionalities. This section first addresses the categories of involved actors, and then introduces a series of use cases representing the functional requirements of our designed system.

In Section 4.2.1, different types of stakeholders are defined. In 4.2.2 a summary of all use cases is presented, following by 20 specific use case definitions for the product specification system.

4.2.1 Actors: Stakeholders in the system

The term stakeholder refers to a person, a group of people, or an organization/enterprise that has certain direct or indirect interest and involvement in the system. Typically, individual stakeholders may play different roles in the environment, e.g. an EPC may act as a consultant in one case, and as a customer in another case.

In GloNet, stakeholders refer to all organizations involved in different stages of the life cycle of the complex products. The roles of these stakeholders differ widely, and some are very specific to the complex products (e.g. the EPC), while others are not and have a more general function (e.g. the supplier). Nevertheless, identifying the main stakeholder organizations, common among different complex products environments is necessary, in order to design the needed product specification system. Four main types of stakeholders of this environment, who have the key roles in functionality-related use cases, are addressed below in this subsection.

4.2.1.1 EPCs and Project developing firm

The project developing firm and or the EPC are the final providers of the complex product to the customer. The EPC, with the help of the customer and potentially the customer’s consultants, suggests a final configuration of the complex product that is customized to the specific requirements and requests of the customer, and further builds and or supports the operation and maintenance of the final complex product.

4.2.1.2 Suppliers and sellers

The suppliers are the companies that manufacture (e.g. an equipment), and provide (e.g. a manual or software-based service) sub-products. The sellers are the companies that sell different sub-products (Equipment, Business services, and Software based services) related to the complex products. Without the help of the suppliers and their commitment, the EPC is not capable of building and or supporting the operation and maintenance of the complex products.

4.2.1.3 Customers

The customer is the one who pays for the final complex product. He/she has a specific set of requests (requirements and preferences) for the complex product that must be fulfilled. With the help of the EPC and consultants, the customer configures and gradually specifies the complex product that he/she needs, based on these requirements.
4.2.1.4 Consultants

Usually, the consultants are not directly involved in the operation of the complex product. They are either hired and paid by the customer to make sure that the customer receives what he/she wants in best conditions or they are hired by (or are employees of) the government, responsible to control if the complex product fulfills specific regulations. An example of a governmental consultant is the representative of the EEA (European Environmental Agency) that might be for instance asked about the regulations related to the plans for the construction of a new solar power plant.

4.2.2 Use Cases

Each user of our designed system may interact with the system differently, depending on what class of stakeholder it falls under. In general, the users that are classified as customer or consultant (Customer’s consultant) only interact with the product specification system in terms of viewing its data and/or approving data (e.g. offers) made by other stakeholders related to the complex product.

An offer is at the heart of the product specification system. It allows the providers of a sub-product (Suppliers for equipment, or providers of the software-based services and business services), to offer a specific product to the EPC, project developer, or customer), related to certain element of the complex-product. It is important to note that for the product specification system it is assumed that for viewing and evaluating offers, while frequently the EPC member will act alone (on behalf of the customer), it may happen that the customer and the EPC members sit together for configuring the plan for the complex product, and for viewing the offers related to the complex product, at least to check certain criteria, e.g. price, delivery condition, etc. Then the offer maybe approved or rejected by the relevant stakeholders (e.g. the consultants of the customer or the customer).

The use-case diagram in Figure 10 shows the interactions of the supplier, seller, EPC, project developing firm, customer and consultant, with the product specification system. It also indicates the set of system administration interactions with this system.

Please note that in the remaining subsections, the EPC/Project developing firm is simply indicated as EPC. Also, please note that since the same set of use cases below apply to both “sub-products” as well as the “complex product” itself, the name “product” is selected to refer to both of these features. Furthermore, the functions named as “manage”, introduced for different classes in the use cases, refer to a set of manipulation functionalities related to those classes. For instance “manage offers” addresses the: add, edit, view, and change status of the offer class.
Offer related use cases – Use Cases # 1 to 5

4.2.3 Use Case 1 - Managing offers

As noted before, the offer is at the heart of the product specification system. The stakeholders that can create an offer include the suppliers, sellers, and the EPC.
Each supplier/seller is in charge of managing *(add, edit, view)* its own offers, which are the offers that are made by it, (i.e. offers related to sub-products of a complex product). Furthermore, these offers are usually used by the EPC for making a more composite offer related to the complex product. On the other hand, the EPC may be in charge of managing the complete large offer(s) that is made for the entire complex product. In some cases, the EPC may create multiple offers related to big sub-products of the complex product. For example, the EPC may address the construction and the maintenance support of a complex product in 2 separate offers. On the other hand, the EPC can receive and then manage offers made by other stakeholders, although these offers can then only be viewed, approved, or rejected by it.

Finally, the customer (with the advice of the EPC/project developer) and consultants (authorized by the customer) represent the ultimate consuming side of the offers that can only manage offers made by other stakeholders to them, in most cases by an EPC, although these offers can only be viewed, approved, or rejected.

Please note that in principle, within a collaboration community/alliance established among the complex products’ stakeholders, there are some stakeholders who act as providers (e.g. providers of equipment, manual services, software-based services, and composed services, etc.) and the others who act as consumers (e.g. including customers, as well as the intermediary organizations such as EPC, project developer, service integrator, etc.). Within this alliance, different kinds of offer may appear. One kind of offer is related to customization of a sub-product (large or small) to match certain specific requirements related to the complex product. Another kind of offer however, is related to the announcement / advertisement of a sub-product within the alliance. While the former kind of offer is made to a specific recipient, who would be the only stakeholder with the authorization to view and approve/reject that offer, the later kind of offer is made to “public”, and therefore all members of the alliance can view and approve this kind of offer.

Table 1 shows the Use Case of Managing offers.

### Table 1 - Managing Offers Use Case

<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>Manage Offers</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>Supplier/Seller/EPC can review and manage its offers by choosing to add, edit, or view the offer details. EPC/Customer/Consultant can review and manage offers made to them by other stakeholders, by choosing to view, approve or reject the offer.</td>
</tr>
<tr>
<td><strong>Actor</strong></td>
<td>Supplier/Seller/EPC/Customer/Consultant</td>
</tr>
<tr>
<td><strong>Precondition</strong></td>
<td>• Supplier/seller EPC successfully logs into the system</td>
</tr>
<tr>
<td><strong>Normal Flow</strong></td>
<td>1. System displays the offers that the Supplier/Seller/EPC has made 2. System displays the offers that other stakeholders have made to the EPC/Customer/Consultant or to the public. 3. The stakeholder selects an offer to view, edit, approve, reject or selects to add a new offer depending on its role 4. System redirects the user to the relevant use case.</td>
</tr>
<tr>
<td><strong>Alternative Flows</strong></td>
<td></td>
</tr>
</tbody>
</table>
4.2.4 Use Case 2 - Adding a new Offer (Offering a Product)

Any supplier, seller or the EPC can add an Offer. When a stakeholder adds an Offer, he/she is actually announcing that he/she can produce/provide the needed sub-product or complex product with the conditions stated in the offer. When adding an offer, the user also provides one or more object-classes for the offer. These object-classes indicate what features the user provides for the offer and what stakeholders are involved with this Offer.

Table 2 shows the use case of adding a new Offer.

Table 2 - Adding a new Offer Use Case

<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>Add Offer</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>Supplier/Seller/EPC configures and offers a sub-product, or the entire complex product, in case of EPC</td>
</tr>
<tr>
<td><strong>Actor</strong></td>
<td>Supplier/Seller/EPC</td>
</tr>
</tbody>
</table>
| **Precondition** | • Supplier/Seller/EPC successfully logs into the system  
• A Supplier/Seller/EPC has already configured the sub-product or complex product |
| **Normal Flow** | 1. Supplier/Seller/EPC requests to add a new offer  
2. System provides the Supplier/Seller/EPC with the possible organizations to which she/he can make an offer, or to select the public option.  
3. Supplier/Seller/EPC selects an organization to which she/he wants to make the offer, or decides to make the offer public.  
4. System provides the Supplier/Seller/EPC with the possible sub-products or complex products he/she can offer.  
5. Supplier/Seller/EPC selects one or more sub-product or the complex product.  
6. Supplier/Seller/EPC provides the object-classes that the offer falls under  
7. System asks the Supplier/Seller/EPC to provide all features based on the object-classes provided  
8. Supplier/Seller/EPC provides the feature values and adds other new features if needed |
9. System checks the validity of the input
10. System inserts newly entered offer data
11. System returns the unique ID for the new offer

**Alternative Flows**

| a.10. System displays an error message for invalid fields and asks the user to re-enter new values for invalid fields.  
| a.11. User re-enters new values for the invalid information  
| a.12. Go back to step 9 |

### 4.2.5 Use Case 3 - Editing an Offer

After an offer has been made by a stakeholder it might decide to change its offer because of two main reasons: Either the offer has been rejected by another party or a condition has changed in the outside world that has made the provider of the offer (namely the supplier, seller or the EPC) to change his mind.

The provider of the offer can only edit an offer if the stakeholder to which the offer has been made has not yet accepted the offer or in the case of public offers, when no one has yet accepted the offer. Table 3 shows the use case of editing an offer.

<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>Edit (Modify) offer</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>Supplier/Seller/EPC modifies an offer it has made for a product</td>
</tr>
<tr>
<td><strong>Actor</strong></td>
<td>Supplier/Seller/EPC</td>
</tr>
<tr>
<td><strong>Precondition</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Supplier/Seller/EPC successfully logs into the system</td>
</tr>
<tr>
<td><strong>Normal Flow</strong></td>
<td></td>
</tr>
</tbody>
</table>
| 1. Supplier/Seller/EPC requests to edit an offer (by providing the ID of the offer or via the manage offer use case)  
| 2. System retrieves the related offer data and displays them to the Supplier/Seller/EPC  
| 3. Supplier/Seller/EPC modifies the offer  
| 4. System checks the validity of the input  
| 5. System moves the current state of the offer to history  
| 6. System changes the status of offer to “Revised”  
| 7. System updates the offer data |
| **Alternative Flows** |  
| a.2. Because of the invalid ID (either it does not exist or it has not been created by the organization to which the user belong to), system returns an error and asks Supplier/Seller/EPC to re-enter ID information  
| a.3. Go back to step 2  
| b.4. System displays an error message for invalid fields and asks the user to re-enter new values for invalid fields  
| b.5. User re-enters the new values for invalid information  
| b.6. Go back to step 4 |

### 4.2.6 Use Case 4 – Changing the status of an Offer

After a stakeholder has made an offer, she/he can change its status from “New” or “Revised” to “withdrawn”. Furthermore, the status of an offer can be changed to “Accepted” or “Rejected”, by the stakeholder to whom the offer has been made. Please note that for the public offers, any stakeholder that has a customer, consultant or EPC role has the right to change the status of such an offer to “Accepted”. In such a case, the offer will change from a public offer to an offer made to that specific accepting stakeholder.

Table 4 shows the use case for changing the status of the offer.

---

1 Provided through support of versioning, by the GloNet platform [6].
Table 4 – Changing the status of an Offer Use Case

<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>Changing the status of offer</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>Supplier/Seller/EPC/Customer/Consultant changes the status of the offer made for a product.</td>
</tr>
<tr>
<td><strong>Actor</strong></td>
<td>Supplier/Seller/EPC/Customer/Consultant</td>
</tr>
<tr>
<td><strong>Precondition</strong></td>
<td>Supplier/Seller/EPC/Customer/Consultant successfully logs into the system</td>
</tr>
</tbody>
</table>
| **Normal Flow** | 1. Supplier/Seller/EPC/Customer/Consultant requests a status change for the (by providing the new status and the ID of the offer or via the manage offer use case)  
2. System checks if the stakeholder is allowed and new status is valid.  
3. System moves the current state of the offer to history  
4. System changes the status of the offer |
| **Alternative Flows** | a.2. Because of the invalid ID (it does not exist), system returns an error and asks Supplier/Seller/EPC/Customer/Consultant to re-enter ID information  
a.3. Go back to step 2  
b.3. Because the new status is invalid or the user is not allowed to set the status, system return an error and asks for another status  
b.4. Go back to step 2 |

4.2.7 Use Case 5 - Viewing an Offer

After an offer has been made, it becomes available for the offering stakeholder as well as either for the specific stakeholder to whom the offer has been made, or in the case of a public offer for all stakeholders with a customer, consultant, or EPC role, for the viewing purposes. Table 5 shows the use case for viewing an Offer.

Table 5 - Viewing an Offer Use Case

<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>View offer</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>The user can view the details of an offer and accept or reject the offer</td>
</tr>
<tr>
<td><strong>Actor</strong></td>
<td>Supplier/Seller/ EPC/Customer/Consultant</td>
</tr>
<tr>
<td><strong>Precondition</strong></td>
<td>The user successfully logs into the system</td>
</tr>
</tbody>
</table>
| **Normal Flow** | 1. The user enters the ID of the offer  
2. System retrieves the related offer data and displays them to the user |
| **Alternative Flows** | a.2. Because of the invalid ID, system returns an error and asks the user to re-enter the ID information  
a.3. Go back to step 2 |

Product related use cases – Use Cases # 6 to 9

4.2.8 Use Case 6 - Managing Products

Before an offer can be made, the product must be defined. A product can be either the complete complex product defined by the EPC, or a sub-product defined by a supplier or seller. The EPC/supplier/seller can view and duplicate all available products independent of who made the product initially but no one can edit or remove a product after it is added. Table 6 shows the use case of managing products.

---

2 Provided through support of versioning, by the GloNet platform [6].
### Table 6 - Managing Products Use Case

<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>Manage Products</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>Supplier/Seller/EPC can review all products and can choose to add, duplicate, or view a product from among all available products.</td>
</tr>
<tr>
<td><strong>Actor</strong></td>
<td>Supplier/Seller/EPC</td>
</tr>
<tr>
<td><strong>Precondition</strong></td>
<td>• Supplier/Seller/EPC successfully logs into the system</td>
</tr>
<tr>
<td><strong>Normal Flow</strong></td>
<td>1. System displays all available products</td>
</tr>
<tr>
<td></td>
<td>2. Supplier/Seller/EPC selects a product to view or duplicate, or selects to add a new product</td>
</tr>
<tr>
<td></td>
<td>3. System redirects the user to the relevant use case</td>
</tr>
</tbody>
</table>

### Alternative Flows

![Product Use Case diagram](image)

#### 4.2.9 Use Case 7 - Adding a new Product

Any supplier, seller or EPC can define (add) a new product. When adding a new product it must also define (or identify if existing) the sub-products, and define (or identify if existing) object-classes of which it falls under. Furthermore, the object-classes require the user to provide a set of features for the product, which make it possible to compare different products from the same set of object-classes. Object-classes also provide the option for stakeholders to filter their search on a specific object-class, or set of object-classes.

Table 7 shows the use case of adding a new Product.

<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>Add Product</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>Supplier/Seller/EPC configures a product</td>
</tr>
<tr>
<td><strong>Actor</strong></td>
<td>Supplier/Seller/EPC</td>
</tr>
<tr>
<td><strong>Precondition</strong></td>
<td>Supplier/Seller/EPC successfully logs into the system</td>
</tr>
<tr>
<td><strong>Normal Flow</strong></td>
<td>1. Supplier/Seller/EPC requests to add a new product</td>
</tr>
<tr>
<td></td>
<td>2. Supplier/Seller/EPC provides the object-classes that the product falls under</td>
</tr>
<tr>
<td></td>
<td>3. Supplier/Seller/EPC selects a set of sub-products that together constitute the product</td>
</tr>
<tr>
<td></td>
<td>4. System asks the Supplier/Seller/EPC to provide the required features, based on the features defined for its object-classes.</td>
</tr>
</tbody>
</table>
5. Supplier/Seller/EPC provides the features and may add additional features if needed
6. System checks the validity of the input
7. System inserts newly entered product data
8. System returns the unique ID for the new product

**Alternative Flows**

5. System displays an error message for invalid fields and asks the user to re-enter new values for invalid fields.
6. User re-enters the new values for invalid information
7. Go back to step 6

### 4.2.10 Use Case 8 – Duplicating and Editing a Product

After a product has been defined (or added), the product can be duplicated, so that its definition can be edited. Clearly stakeholders cannot edit the definition of an existing (already introduced) product, but they can of course edit a duplicate of it, if the product they are introducing is somewhat different from an existing product. This decision has been made in the design of our product specification system to guarantee that the history of product definitions are preserved and to prohibit editing a product’s specifications while certain offers, or certain other composite products maybe referring to them. Table 8 shows the use case of duplicating and editing a product.

<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>Duplicate and edit Product</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>Supplier/Seller/EPC duplicates and modifies available products</td>
</tr>
<tr>
<td><strong>Actor</strong></td>
<td>Supplier/Seller/EPC</td>
</tr>
<tr>
<td><strong>Precondition</strong></td>
<td>Supplier/Seller/EPC successfully logs into the system</td>
</tr>
<tr>
<td><strong>Alternative Flows</strong></td>
<td>a.2. Because of the invalid ID (it doesn’t exist), system returns an error and asks Supplier/Seller/EPC to re-enter ID information  a.3. Go back to step 2  b.4. System displays an error message for invalid fields and asks the user to re-enter new values for invalid fields  b.5. User re-enters new values for the invalid information  b.6. Go back to step 4</td>
</tr>
</tbody>
</table>

### 4.2.11 Use Case 9 - Viewing a Product

After a product has been defined (added), it can be viewed by stakeholders that either have an EPC, Seller or Supplier role. Viewing the product provides the details about the product. Table 9 shows the use case of viewing a product.

<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>View Product</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>The user can view the details of a product</td>
</tr>
<tr>
<td><strong>Actor</strong></td>
<td>Supplier/Seller/EPC</td>
</tr>
</tbody>
</table>
**Object-class related use cases – Use Cases # 10 to 13**

4.2.12 Use Case 10 - Managing Object-classes

Object-classes have an important role in the product specification system. When defining a Product or an offer, they enforce the user to provide the feature details (meta-data) that fully characterize the product or offer. The Suppliers, Sellers and the EPC can manage *(add, duplicate and edit, and view)* all available object-classes. Therefore when applicable, these users can benefit and reuse the existing definition (using the duplicate and edit function) of other object classes (potentially added by other stakeholders) as the base.

Table 10 shows the use case for managing object-classes.

<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>Manage Object-classes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>Supplier/Seller/EPC can review all available object-classes for both products and product offers and can choose to add a new object-class, edit an object-class that is added by him/her, or view an object-class from all existing object-classes in the system</td>
</tr>
<tr>
<td><strong>Actor</strong></td>
<td>Supplier/Seller/EPC</td>
</tr>
<tr>
<td><strong>Precondition</strong></td>
<td>• Supplier/Seller/EPC successfully logs into the system</td>
</tr>
<tr>
<td><strong>Normal Flow</strong></td>
<td>1. System displays all available object-classes</td>
</tr>
<tr>
<td></td>
<td>2. Supplier/Seller/EPC selects an object-class to view or edit or selects to add a new object-class</td>
</tr>
<tr>
<td></td>
<td>3. System redirects the user to the relevant use case</td>
</tr>
<tr>
<td><strong>Alternative Flows</strong></td>
<td></td>
</tr>
</tbody>
</table>
4.2.13 Use Case 11 - Adding a new Object-class

An EPC, Supplier or Seller can add a set of object-classes. These object-classes are used for defining both the product and the offer. When defining (adding) a new object-class, the user must also define the feature-kinds for the object-class. These feature-kinds are the set of attributes that together characterize an object-class, and for which their instances’ features have values. Table 11 shows the use case for adding a new object-class.

<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>Add Object-class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Supplier/Seller/EPC configures an object-class</td>
</tr>
<tr>
<td>Actor</td>
<td>Supplier/Seller/EPC</td>
</tr>
<tr>
<td>Precondition</td>
<td>• Supplier/Seller/EPC successfully logs into the system</td>
</tr>
</tbody>
</table>
| Normal Flow   | 1. Supplier/Seller/EPC requests to add a new object-class  
2. Supplier/Seller/EPC provides a name for the object-class  
3. Supplier/Seller/EPC selects a set of feature-kinds for the object class that characterize the product or the offer that object-class represents.  
4. System checks the validity of the input  
5. System inserts the newly entered object-class  
6. System returns an ID for the new object-class |
| Alternative Flows | a.4. System displays an error message for invalid fields and asks the user to re-enter new values for invalid fields.  
a.5. User re-enters new values for the invalid information  
a.6. Go back to step 4 |

4.2.14 Use Case 12 – Duplicating and editing an existing Object-class

After an object-class has been added, the stakeholders can duplicate and edit the duplicate. Table 12 shows the use case of editing an existing object-class.

<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>Duplicate and edit Object-class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Supplier/Seller/EPC duplicates and modifies object-classes available in the system.</td>
</tr>
<tr>
<td>Actor</td>
<td>Supplier/Seller/EPC</td>
</tr>
<tr>
<td>Precondition</td>
<td>• Supplier/Seller/EPC successfully logs into the system</td>
</tr>
</tbody>
</table>
| Normal Flow   | 1. Supplier/Seller/EPC enters the ID of the object-class  
2. System retrieves the related object-class data and displays them to the Supplier/Seller/EPC  
3. Supplier/Seller/EPC modifies the object-class  
4. System checks the validity of the input  
5. System inserts newly entered object-class data  
6. System returns the unique ID for the new object-class |
| Alternative Flows | a.2. Because of the invalid ID (it doesn’t exist), system returns an error and asks Supplier/Seller/EPC to re-enter ID information  
a.3. Go back to step 2  
a.5. System displays an error message for invalid fields and asks the user to re-enter new values for invalid fields  
b.6. User re-enters new values for the invalid information  
b.7. Go back to step 4 |
4.2.15 Use Case 13 - Viewing an Object-class

After an object-class has been defined (added), it becomes available for the stakeholders that have an EPC, Supplier or Seller role for viewing purposes. Viewing the Object-class provides its detailed description.

Table 13 shows the use case of viewing a product.

<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>View Object-class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>The user can view the details of an object-class</td>
</tr>
<tr>
<td>Actor</td>
<td>Supplier/Seller/EPC</td>
</tr>
<tr>
<td>Precondition</td>
<td>• The user has successfully logs into the system</td>
</tr>
</tbody>
</table>
| Normal Flow   | 1. The user enters the ID of the object-class  
                2. System retrieves the related object-class data and displays it to the user |
| Alternative Flows | a.1. Because of the invalid ID, system returns an error and asks the user to re-enter ID information  
                a.2. Go back to step 1 |

4.2.16 Use Case 14 - Managing Feature-kind

Before a user can define (add) an Object-class, the feature-kinds of that object-class must have been first defined. The user can manage *(add, duplicate, edit, and view)* all available feature-kind. A feature-kind is a tuple including a name, the type of data the feature-kind contains (its domain) and the scales it can have.

Table 14 shows the use case of managing feature-kinds.

<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>Manage Feature-kinds</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Supplier/Seller/EPC can review all available feature-kinds and can choose to add a new feature-kind, duplicate and edit a feature-kind that is available in the system or view a feature-kind from all available feature-kinds.</td>
</tr>
<tr>
<td>Actor</td>
<td>Supplier/Seller/EPC</td>
</tr>
<tr>
<td>Precondition</td>
<td>• Supplier/Seller/EPC successfully logs into the system</td>
</tr>
</tbody>
</table>
| Normal Flow   | 1. System displays the name of all available feature-kinds  
                2. Supplier/Seller/EPC selects a feature-kind to view or duplicate and edit, or selects to add a new feature-kind  
                3. System redirects the user to the relevant use case |
| Alternative Flows | |
4.2.17 Use Case 15 - Adding a new Feature-kind

Any EPC, Supplier or Seller can add a set of feature-kinds into the system. These feature-kinds are used during the process of adding an object-class. When defining (adding) a new feature-kind, the user provides the type and possible scales that a feature-kind can have. Table 15 shows the use case for adding a new feature-kind.

Table 15 – Adding a Feature-kind Use Case

<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>Add Feature-kind</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Supplier/Seller/EPC configures a feature-kind</td>
</tr>
<tr>
<td>Actor</td>
<td>Supplier/Seller/EPC</td>
</tr>
<tr>
<td>Precondition</td>
<td>Supplier/Seller/EPC successfully logs into the system</td>
</tr>
</tbody>
</table>
| Normal Flow   | 1. Supplier/Seller/EPC requests to add a new feature-kind  
                          2. Supplier/Seller/EPC provides the name and type of feature-kind.  
                          3. Supplier/Seller/EPC selects a set of possible scales for the feature-kind, when applicable.  
                          4. System asks the Supplier/Seller/EPC to provide sub-feature-kinds if the type of the feature-kind is composed.  
                          5. Supplier/Seller/EPC selects a set of sub-feature-kinds that together form the feature-kind (If the type is composed).  
                          6. System checks the validity of the input  
                          7. System inserts data related to newly entered feature-kind  
                          8. System returns a unique ID for the new feature-kind |
| Alternative Flows | a.7. System displays an error message for invalid fields and asks the user to re-enter new values for invalid fields.  
                          a.8. User re-enters new values for the invalid information  
                          a.9. Go back to step 6 |

4.2.18 Use Case 16 – Duplicating and Editing a Feature-kind

After a stakeholder has added a feature-kind, any other stakeholder with the roles supplier, seller or EPC can duplicate it and edit the duplicate. Table 16 shows the use case of editing an existing feature-kind.
Table 16 – Duplicating and editing a Feature-kind Use Case

<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>Duplicate and edit Feature-kind</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Supplier/Seller/EPC duplicates and modifies feature-kinds available in the system</td>
</tr>
<tr>
<td>Actor</td>
<td>Supplier/Seller/EPC</td>
</tr>
<tr>
<td>Precondition</td>
<td>• Supplier/Seller/EPC successfully logs into the system</td>
</tr>
<tr>
<td>Normal Flow</td>
<td>1. Supplier/Seller/EPC enters the ID of the feature-kind</td>
</tr>
<tr>
<td></td>
<td>2. System retrieves the related feature-kind data and displays it to the Supplier/Seller/EPC</td>
</tr>
<tr>
<td></td>
<td>3. Supplier/Seller/EPC modifies the feature-kind</td>
</tr>
<tr>
<td></td>
<td>4. System checks the validity of the input</td>
</tr>
<tr>
<td></td>
<td>5. System adds the new feature-kind data</td>
</tr>
<tr>
<td></td>
<td>6. System returns the unique id for the new feature-kind</td>
</tr>
<tr>
<td>Alternative Flows</td>
<td>a.2. Because of the invalid ID (if it does not exist), system returns an error and asks Supplier/Seller/EPC to re-enter new ID information</td>
</tr>
<tr>
<td></td>
<td>a.3. Go back to step 2</td>
</tr>
<tr>
<td></td>
<td>b.4. System displays an error message for invalid fields and asks the user to re-enter new values for invalid fields</td>
</tr>
<tr>
<td></td>
<td>b.5. User re-enters new values for the invalid information</td>
</tr>
<tr>
<td></td>
<td>b.6. Go back to step 4</td>
</tr>
</tbody>
</table>

4.2.19 Use Case 17 - Viewing a Feature-kind

After a feature-kind has been defined (added) it becomes available for the stakeholders that have an EPC, supplier or seller role for viewing. Viewing the feature-kind provides details that characterize its meta-data.

Table 17 shows the use case of viewing a feature-kind.

Table 17 – Viewing a Feature-kind Use Case

<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>View Feature-kind</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>The user can view the details of a feature-kind</td>
</tr>
<tr>
<td>Actor</td>
<td>Supplier/Seller/EPC</td>
</tr>
<tr>
<td>Precondition</td>
<td>• The user successfully logs into the system</td>
</tr>
<tr>
<td>Normal Flow</td>
<td>1. The user enters the ID of the feature-kind</td>
</tr>
<tr>
<td></td>
<td>2. System retrieves the related feature-kind data and displays it to the user</td>
</tr>
<tr>
<td>Alternative Flows</td>
<td>a.2. Because of the invalid ID, system returns an error and asks the user to re-enter ID information</td>
</tr>
<tr>
<td></td>
<td>a.3. Go back to step 1</td>
</tr>
</tbody>
</table>

Scales related use cases – Use Cases # 18 to 21

4.2.20 Use Case 18 - Managing Scales

Before a Supplier/Seller/EPC can define (add) a Feature-kind, the Scales that a feature-kind supports must be defined in the system. The Supplier/Seller/EPC can manage (add, duplicate and edit, and view) all available scales, e.g. Centimetres, Celsius, etc.

Table 18 shows the use case of managing scales.

---

3 Provided through the support of authorized access, by the GLONET platform [6]
<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>Manage Scales</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Supplier/Seller/EPC can review all available scales and can choose to add a new scale, edit a scale that is added by him/her, or view a scale from all available scales</td>
</tr>
<tr>
<td>Actor</td>
<td>Supplier/Seller/EPC</td>
</tr>
<tr>
<td>Precondition</td>
<td>• Supplier/Seller/EPC successfully logs into the system</td>
</tr>
<tr>
<td>Normal Flow</td>
<td>1. System displays all available scales by name</td>
</tr>
<tr>
<td></td>
<td>2. Supplier/Seller/EPC selects a scale to view or edit, or selects to add a new scale</td>
</tr>
<tr>
<td></td>
<td>3. System redirects the user to the relevant use case</td>
</tr>
</tbody>
</table>

4.2.21 Use Case 19 - Adding a new Scale

Any EPC, Supplier or Seller can add a set of Scales. These Scales are used during the process of adding a Feature-kind. When defining (adding) a new Scale, the user provides a name for the Scale. Table 19 shows the use case for adding a new Scale.

<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>Add Scale</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Supplier/Seller/EPC configures a scale</td>
</tr>
<tr>
<td>Actor</td>
<td>Supplier/Seller/EPC</td>
</tr>
<tr>
<td>Precondition</td>
<td>• Supplier/Seller/EPC successfully logs into the system</td>
</tr>
<tr>
<td>Normal Flow</td>
<td>1. Supplier/Seller/EPC requests to add a new scale</td>
</tr>
<tr>
<td></td>
<td>2. Supplier/Seller/EPC provides the name of the scale.</td>
</tr>
<tr>
<td></td>
<td>3. System checks the validity of the input</td>
</tr>
<tr>
<td></td>
<td>4. System inserts newly entered scale data</td>
</tr>
<tr>
<td></td>
<td>5. System returns a unique ID for the new scale</td>
</tr>
<tr>
<td>Alternative Flows</td>
<td>a.3. System displays an error message for invalid fields and asks the user to re-enter new values for invalid fields.</td>
</tr>
<tr>
<td></td>
<td>a.4. User re-enters new values for the invalid information</td>
</tr>
<tr>
<td></td>
<td>a.5. Go back to step 3</td>
</tr>
</tbody>
</table>
4.2.22 Use Case 20 – Duplicating and Editing a Scale

After a stakeholder has added a scale, any other stakeholder with the roles supplier, seller or EPC can duplicate it and edit the duplicate.

Table 20 shows the use case of editing an existing Scale.

Table 20 – Duplicating and editing a Scale Use Case

<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>Duplicate and edit Scale</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Supplier/Seller/EPC duplicates and modifies scales available in the system</td>
</tr>
<tr>
<td>Actor</td>
<td>Supplier/Seller/EPC</td>
</tr>
<tr>
<td>Precondition</td>
<td>• Supplier/Seller/EPC successfully logs into the system</td>
</tr>
<tr>
<td>Normal Flow</td>
<td>1. Supplier/Seller/EPC enters the ID of the scale</td>
</tr>
<tr>
<td></td>
<td>2. System retrieves the related scale data and displays them to the Supplier/Seller/EPC</td>
</tr>
<tr>
<td></td>
<td>3. Supplier/Seller/EPC modifies the scale</td>
</tr>
<tr>
<td></td>
<td>4. System checks the validity of the input</td>
</tr>
<tr>
<td></td>
<td>5. System adds the new scale data</td>
</tr>
<tr>
<td></td>
<td>6. System returns the unique id for the new scale</td>
</tr>
<tr>
<td>Alternative Flows</td>
<td>a.2. Because of the invalid ID (if it does not exist), system returns an error and asks</td>
</tr>
<tr>
<td></td>
<td>Supplier/Seller/EPC to re-enter new ID information</td>
</tr>
<tr>
<td></td>
<td>a.3. Go back to step 2</td>
</tr>
<tr>
<td></td>
<td>b.4. System displays an error message for invalid fields and asks the user to re-enter</td>
</tr>
<tr>
<td></td>
<td>new values for invalid fields</td>
</tr>
<tr>
<td></td>
<td>b.5. User re-enters new values for the invalid information</td>
</tr>
<tr>
<td></td>
<td>b.6. Go back to step 4</td>
</tr>
</tbody>
</table>

4.2.23 Use Case 21 - Viewing a Scale

After a Scale has been defined (added), it becomes available for the stakeholders that have an EPC, Supplier or Seller role for viewing. Viewing the Scale provides the related details about the scale.

Table 21 shows the use case of viewing a Scale.

Table 21 - Viewing a Scale Use Case

<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>View Scale</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>User can view details of a scale</td>
</tr>
<tr>
<td>Actor</td>
<td>Supplier/Seller/EPC</td>
</tr>
<tr>
<td>Precondition</td>
<td>• The user successfully logs into the system</td>
</tr>
<tr>
<td>Normal Flow</td>
<td>1. The user enters the ID of the scale</td>
</tr>
<tr>
<td></td>
<td>2. System retrieves the related scale data and displays them to the user</td>
</tr>
<tr>
<td>Alternative Flows</td>
<td>a.2. Because of the invalid ID, system returns an error and asks the user to re-enter</td>
</tr>
<tr>
<td></td>
<td>new ID information</td>
</tr>
<tr>
<td></td>
<td>a.3. Go back to step 1</td>
</tr>
</tbody>
</table>
5. DATABASE DESIGN

This section addresses the design of the meta-data (schema) of the database needed for the implementation of our product specification system. Database systems traditionally follow the relational, object-oriented, or object-relational data models, while the majority of existing database systems are relational.

5.1 Relational database schema design

The system design provided for our product specification system has applied the object-oriented approach. In fact this decision was made, in order to capture the required design flexibility and to provide reusability for the design concepts and mechanisms. Nevertheless, for the database development purpose, based on the fact that most software platforms are well supported for wrapping and providing tools for relational databases, we have opted for a relational database implementation. Therefore, the design of our database schema is based on the relational approach and uses well-known tools, such as the MySQL Workbench.

Based on the product specification system design addressed in this deliverable, for supporting the definition of the identified entities and their relationships involved in complex products in GloNet, we have designed the following relational database schema that preserves its addressed requirements. Figure 16 shows the corresponding relational schema, which is designed using the MySQL Workbench.

![Relational Database Schema](image_url)

Figure 16. Relational Database Schema for the Product Specification System
6. USER INTERACTION AND INTERFACES DESIGN

This section aims to present the design of data manipulation services which are planned to be developed for the product specification system, as well as to give a feeling for how the implemented interface will look like.

6.1 User interface functionality design
While the implementation of the product specification system is the next step of this work, at this stage we have also designed an abstract view for the functionality (data manipulation services), which will be provided through this user interface. In other words, clearly the user interface for the product specification system would be user-friendly and easy to understand, while hiding the complexity of this system. But at the moment, the aim of this design is to only to give a first feeling for how the planned functionality can be used through this interface. Figure 17 shows how the abstract user interfaces will fit into the general GloNet framework user interface [6][10]. Please note that in order to avoid repetition, in the following sub-sections, the designed frames only address the interface to the “Add” function, while the “Edit”, “View”, and “Duplicate” interfaces would be quite similar in principle.

Figure 17. Positioning of the abstract user interface within the GloNet user interface

6.1.1 Product
Figure 18 shows an interface window for introducing a new product. This window is used to add any new sub-product or the complex product itself. To simplify the presentations in this section, this window is called “New Product” window. In this window, the user can add a product by first providing the unique name of the product. The user can also optionally provide one or more object-classes, sub-products and/or features relevant to this new product, for which their definitions are addressed in
details in previous sections. It is important to note that when the user adds a new object-class, some features might be added for it and made mandatory automatically. For example, in Figure 18 since the user has added the object-class “electrical”, the feature with the feature-kind “input voltage” has been made obligatory for it, as seen below. If the added product is composed and thus includes some other sub-products, then those can be also defined. To add a sub-product, the user must also indicate the quantity for the sub-product. For example, a power box might have 2 switches and in this case a sub-product of the power box is the “switch” with the quantity 2. Finally, in order to add a feature to the added product, the user should first indicate its feature-kind, to which the feature corresponds (i.e. the Output voltage), and then the value for the feature, and the scale of the feature, as represented below.

![New Product window](image)

**Figure 18. New Product window**

### 6.1.2 Offer

The communication between the providers of sub-products and their consumers is formalized through offers. In order to make a new offer, the user will use the new offer window. Figure 19 shows an abstract view of this window. To simplify the presentations in this section, this window is called “New Offer” window. In this window, the user must choose one or more products to offer, and the default value for the recipient of the offer is “public” (as explained before). It is not obligatory to provide any other information, however the users can indicate one or more object-classes related to the offer. These object-classes might then enforce certain obligatory features, such as the price (enforced by selecting the object-class Final) for the offer. The user can also add further optional features to the offer, as desired, through the use of add more features.
6.1.3 Feature-kind

As discussed earlier, each complex product, sub-product, or offer is characterized by a set of feature-kinds. The “New Feature-kind” window, as indicated in Figure 20 shows an abstraction of this function, and supports the definition and adding of a new feature-kind. In order to add a new feature-kind, the user has to provide a **unique name** for it, as well as the data type for the feature-kind. The user can also optionally add a number of sub-feature-kinds if the feature-kind is composed, and also indicate possible scales for the feature-kind.

6.1.4 Object-class

As discussed before, each product (i.e. complex product, or sub-product, as explained before) or offer could be associated to one or more object-classes. Furthermore, object-classes define the meta-data for products and offers (and in turn the set of feature-kinds that characterizing those classes). Figure
21 shows an abstraction of the “New Object-class” window, which makes the adding of a new object-class possible. To add a new object-class, the user has to provide a unique name for the object-class. The user can also optionally add the obligatory feature-kinds for the products or offers that are associated with this object-class.

![Figure 21. New Object-class window](image)

### 6.1.5 Scale

Users can introduce a new scale for the features in a feature kind. Figure 22 shows an abstraction of the “New Scale” window for this purpose. The user is only required to enter a unique name for the scale in order to create it. These scales can then be defined for the features introduced in different feature-kinds.

![Figure 22. New Scale window](image)
In order to provide customizable configuration of complex products in different domains, such as those introduced in the GloNet project (PV power plants, intelligent buildings, and future incubators), we have designed a complex product specification system. This system introduces generic constructs and functionality to support different complex product’s stakeholders with their customization tasks. It further aims to properly authorize the stakeholders depending on their roles and responsibilities in relation to the product specification.

The provided class diagram represents our proposed generic meta-data for this system. Furthermore, the functional and non-functional requirements of the system are analysed and a set of 21 use cases are defined capturing different functional aspects of the system. Next step would focus on the implementation of the database system and developing the software services supporting the planned functionality and a user-friendly interface for this system.
8. REFERENCES


CONSORTIUM

CAS Software AG, Germany
Project coordinator: Dr. Bernhard Koelmel

UNINOVA – Instituto de Desenvolvimento de Novas Tecnologias, Portugal
Technical coordinator: Prof. Luis M. Camarinha-Matos

Universiteit van Amsterdam, Netherlands

iPLON GmbH The Infranet Company, Germany

Steinbeis GmbH & Co., Germany

SKILL Estrategia S.L., Spain

Komix s.r.o., Czech Republic

Prolon Control Systems, Denmark

Member of the:

www.glonet-fines.eu