

Methodology for the Specification of Software Requirements for an integrated Logistic Platform

Lucio Tirone, Gaetano D'Altrui

Aster S.p.A.

via Tiburtina 1166, 00156 Rome (Italy)

lucio.tirone@aster-te.it

gaetano.daltrui@aster-te.it

Copyright © held by the authors.

Abstract — This article describes a methodology for the specification of the software requirements of an Open Logistic Services Platform. A business processes analysis, through BPMN models of information exchange between private and institutional actors linked to different logistic cycles, has allowed to identify and analyze Use Cases and to build, in SysML language, the Sequence Diagrams that describe the interactions between users. Through MBSE approach, functional requirements able to ensure the processes proper implementation have been derived. Through the UML language, the software architectural design has been provided and, by defining components and artifacts, the software requirements of each module composing the Platform software have been specified.

Keywords — *logistic platform, business process, software component, BPMN, SysML, UML.*

I. INTRODUCTION: THE LIMS PROJECT

The present article shows the application of the Systems Engineering approach to the specification of the software requirements of an integrated logistic platform for the exchange of business information between the users inside the port community, designed and developed within the LIMS (Logistic Information Management Service) project, funded by the Italian Ministry of Research and University. The main objective of the LIMS project is the development and demonstration of a regional-scale Open Services Platform, which allows, on the basis of a unified model of the logistic processes, the harmonized management, and efficient use of commercial digitized information from the relevant authorities and the public and private users operating in the integrated port and interport logistics. In particular, the LIMS platform has been developed considering the B2B (Business to Business) component that defines the commercial transactions between non-institutional users. To this aim, the information flows requested by the various actors involved and the most suitable interfaces to accomplish the exchange of information have been both defined. The project fits into the broader picture of the digitization process of the Public Administration and the rationalization and standardization of transport management at national level, in order to experience, at pilot level in the Campania Region, the B2B service component of the future National Single Window for maritime transport to be implemented in the European Directive 2010/65 by 2015 at the

national and institutional levels. The LIMS platform will represent a web application offering to Port Community users the following key services:

- **Logistic Data Validation Service:** Ship Cycle Services managed by the Port Informer; definition of a harmonized model for the management of the operative stages of ships in port, from the ship announcement to the arrival in port, maneuvers needed for commercial operations, and departure of the ship.
- **Import Goods Management Service:** Management of Customs clearance at sea, sharing of goods customs status as declared by competent authorities. Freight Services: dedicated to the management of logistics operations that relate to the transport services scheduling and delivery of goods.
- **Export Goods Management Service:** Business Services: dedicated to the management of processes related to the booking of goods transport across the ship and its payments; Customer/Hauler Order management; Freight tracking through all transit logistic nodes; Notification of arrival export cargo.

The development of the platform into its Hardware / Software components relies on the generation of a harmonized model that allows the study of solutions related to all aspects of information and documents exchange (information flows) between the multitude of actors involved in the logistic process which may be different for the various Italian Port and Interport systems. In both cases, the platform is the only interface with which users can turn to for accessing to data associated to their own shipment. In the specific case of the Campania Region (in the southern part of Italy), the logistics supply chain primary centers are configured as intermodal road-sea (port of Naples and Salerno) and road-rail (interport of Nola and Marcianise). This model is able to represent the different cycles of logistics processes, depending on the exchange of messages among actors in the port / interport community and based on the info-telematic systems they use. Once completed, it provides all the elements to identify, define and develop the B2B (Business to Business) Services offered to the actors involved in the logistics chain.

Throughout the LIMS project, the Consortium, through the lead of ASTER, has followed the Systems Engineering methodologies. Systems Engineering fosters the definition of common terms, interfaces management and control of those independent elements/systems, while providing controls to ensure that the overall product satisfies the stakeholder needs. It is often not straightforward to understand when a complex system crosses the boundary to become a system of systems, however it is often the case that the development of such an entity is a collaborative challenge in which the several involved teams are charged to develop elements that cooperate across many iterative stages to provide the required capabilities.

II. OVERALL METHODOLOGY

The proposed approach, reported in Fig. 1, is a top-down iterative process that follows the entire typical flow-down of requirements, starting from the acquisition of the user/stakeholder needs, which are elaborated for the definition of the stakeholder requirements, then modeling the processes in a standard format, and proceeding with the definition of the system requirements. The interaction between layers is based on a “customer-supplier” paradigm, according to which each layer interacts with the next layers ‘providing’ a specification of the requested behavior, in the same way as the final user provides its needs to the system developer. The process will end when the set of requirements is well-suited to be developed by a single team. The final output of LIMS Project is a web-based platform, an info-teleomatics system for logistic information exchange used by different actors, interfaced with external institutional systems. Hence, it is useful to model the platform as a system, i.e. an artifact consisting of blocks that, together, pursue a goal [3]. A block can be generally software, hardware, an individual, or any other unit. But the LIMS platform, in its final realization, is a software with a front-end, for user access, and a back-end for the management and implementation of logistic processes, data storage and notification transmission among the involved operators. So, a system model is useful to derive the system requirements, whereas a model of the software, through UML concepts, is needed to represent the different software blocks, their interactions and the requirements allocation. In order to meet the operational requirements described above, the platform must be designed in such a way as to implement efficiently the logistic cycles according to the typical business models of the port and inter-port context. To this aim, the physical and functional architecture, and the provided services of the Platform, have been defined using a Service-Oriented Architecture (SOA) approach, which can support the use of web services to ensure interoperability between different systems. The definition of a system using a SOA approach is composed of the following phases:

- Implementation of a system Model on the basis of the Model Based System Engineering (MBSE) methodology, which allows to define the functionalities and physical composition of a system.
- Services Definition with a Service-Oriented methodology, that suggests to divide and classify a complex problem into smaller and non-merged

problems, and conceives reusable, combinable and available services in a standardized manner.

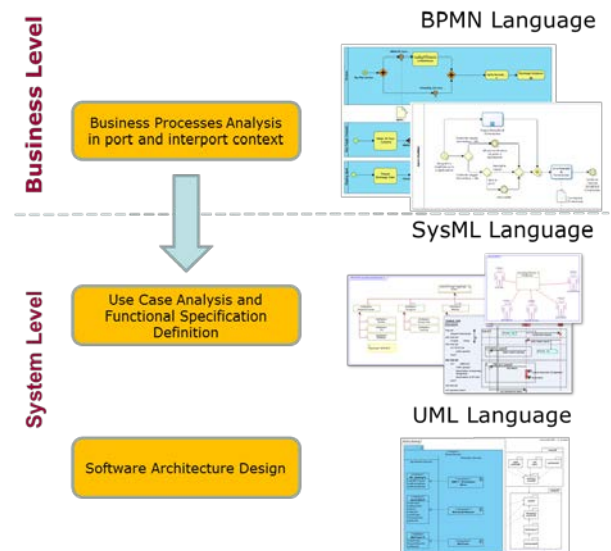


Fig. 1. LIMS Overall Approach

The first step in the realization of the platform model is the analysis of the Business Model, in order to define the information flows and the exchange of data and documents between the various actors involved in the logistic cycles related to ships and goods management. The definition of the Business Model requires a focus on the system context (its boundaries, external actors, external interfaces). The system context diagram shows the system’s environment and thus the system boundary. It is not a predefined diagram of SysML or UML, but a variant of block diagram. In the center of the diagram is the system under development. It is a block with the stereotype *system*. This clearly distinguishes this block from other system blocks yet to be identified. All currently known interaction partners are denoted all around the system and associations are used to connect them

The context of the LIMS platform is represented in Fig. 2. The upper part of the figure indicates the Actors that interact with the system, while the right section illustrates external systems exchanging information with the LIMS Platform. An actor is not a concrete system or a concrete individual, but a role. The system Actors, within LIMS project, are listed below:

- Port Informer;
- Shipping Line (or the Ship Agents, which is the Shipping Line representative);
- Freight Forwarder;
- Terminal Operator;
- Road Hauler;
- Consignee;
- Shipper;
- Port Authority.

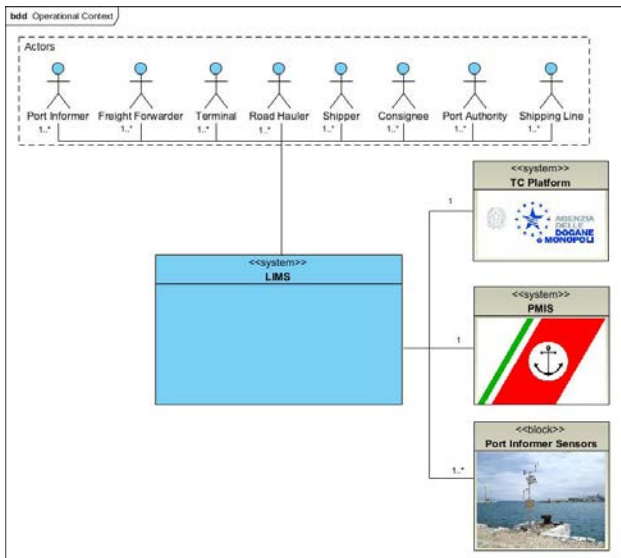


Fig. 2. LIMS Platform Context

The list of External Systems, exchanging information with LIMS Platform, is reported below:

- Port Management Information System (PMIS): it is a system managed by the Italian General Command of Harbour Masters. Informations exchange is related to the Mooring Plan of the Ships;
- TC Platform: it represents the information exchange platform between a terminal operator and the Customs info-telematic system (AIDA-CARGO). LIMS platform requests to TC the Import/Export Goods Manifest, the goods items reference numbers (A3), and the customs status of the goods to be unloaded in a terminal. These information can be accessed with the same temporal rate with which a terminal itself receives them;
- Port Informer Sensors: sensors managed by the Port Informer (AIS portal, meteo-stations, cameras, tide gauges and other sensors network) for the validation of logistic data provided by the Ship Master by means of the radio communications with the Port Informer itself.

The language we choose to formally describe the business process is the BPMN (Business Process Model and Notation), a standard internationally defined by the OMG (Object Management Group). It is able to provide a graphical representation to specify individual processes through a Business Process Diagram (BPD), with a standard, effective and intuitive notation for all the stakeholders involved in the processes. The BPMN diagrams are able to provide a common “framework” upon which it is possible to describe interactions among different operators working in a port or in an interport [1]. The adoption of the BPMN language allows to offer a clear vision of the processes among actors which have heterogeneous characteristics and different responsibilities, also contributing to the modeling of the interactions that take place within the Port Regionalization, where the port cluster is seen as a changing environment to which different logistic

realities are related. A very important feature of the BPMN standard is that it often allows tight integration with software development systems. In fact, applications that allow the BPMN designer to represent the process details using BPMN and then to translate that model into software programs for the process management, are now available. Through dedicated technical meetings with the different operators, the processes that the platform has to implement to guarantee each of the key services defined above, have been defined and validated. These processes have been modeled through the BPMN language, by means of Choreography and Orchestration diagrams. A Choreography diagram, representing the macro-activities of the process represented in a chronological sequence, has been realized for each key service. Each macro-activity is represented by a block showing, in the upper part, the initiating partner, and the secondary actors in the lower part. A dedicated Orchestration diagram has been provided to represent the detailed information exchange between the actors of each block. A section of the BPMN choreography diagram related to the Export Goods Management Services (in particular, the Booking Request task) is reported in Fig. 3 (it represents just one choreography task to which the Orchestration Diagram corresponds).

The orchestration diagram, showing the information flows between the freight forwarder and the hauler for an optimized and digitized (through the platform) management of transport order, is depicted in Fig. 4. The BPMN diagrams show clearly the user tasks, which require the interaction of the user with the platform for the submission of data or the visualization of forms and summary tables, and the send/receive tasks, which indicate the functionality of data structures transmitting to the actors who need the information, adding more data, if needed, to complete the task, for planning purposes and hence speeding up the whole process.

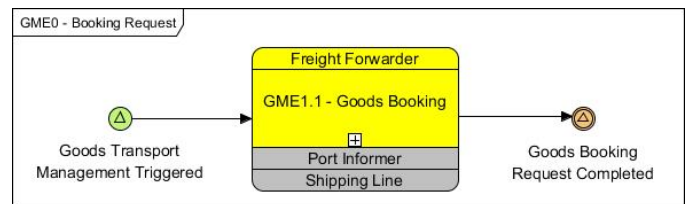


Fig. 3. Choreography Diagram: Export Goods Management Service

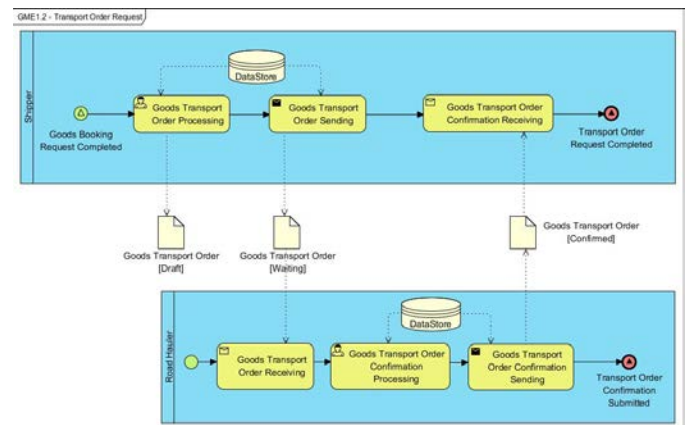


Fig. 4. Orchestration Diagram: Transport Order Management

III. DERIVATION OF PLATFORM REQUIREMENTS

The following step is the identification of the system Use Cases, representing the goals of a system from the perspective of the users, from the analysis of the business processes. Using the SysML language, formal Use Case diagrams are drawn, to show the complete list of actors (primary and secondary), as well as a full text description for each of them to illustrate the goal of the primary actor and the role of the secondary actors. So, the Diagram provides an high-level view of a system functionality, depending on how the actors use the system itself. A typical use case description may include the Preconditions, i.e. the conditions that must hold for the use case to begin, Postconditions, the conditions that must hold once the use case has completed, and the Trigger, which identifies the event that causes the activation of the use case. Fig. 5 shows one example of Use Case Diagram of the case under study, representing the functionality of transport order elaboration related to the process analyzed in Fig. 4. At this point, the analysis of the platform behavior, within each Use Case, can be performed thorough dedicated sequence diagrams, used to describe the main interactions between the system and its environment.

The approach for the derivation of the requirements is based on the detailed analysis of the sequence diagrams, as follows:

- Looking at the Use Case description, draw messages between actors and system. These messages represent an exchange of information rather than data, because the focus is on the functional behavior, independently from the actual physical realization (i.e. different message standards could be used, but the information content of the message is what we are concerned about in this phase).
- Draw a “message to self” before each output message on the system lifeline. These messages correspond to “operations” allocated to the system.

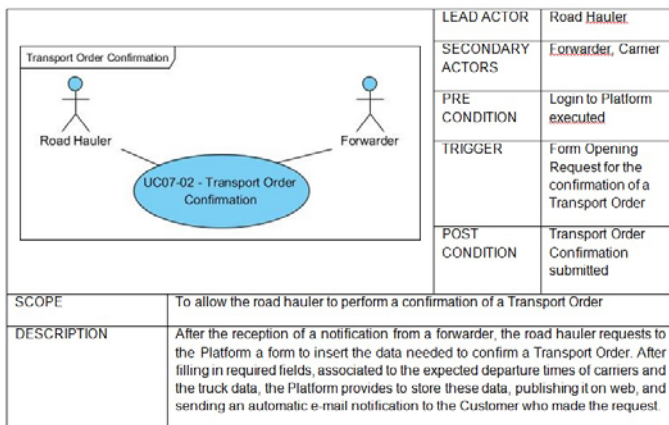


Fig. 5. Use Case Description: Transport Order Confirmation Request

Each operation derived in this way is a very reasonable candidate for a functional requirement, because it has to describe what the system has to do with the input information,

in order to produce the requested output, without any concern on how the system elements interact with each other in the process.

Fig. 6 shows one sequence diagram developed for the case under study, and the requirements derived from it. The first one is related to the request of a form for the declaration of goods data associated to a transport order, while the other two are associated to the request to the platform of saving and submission of the data structure originated from the user form.

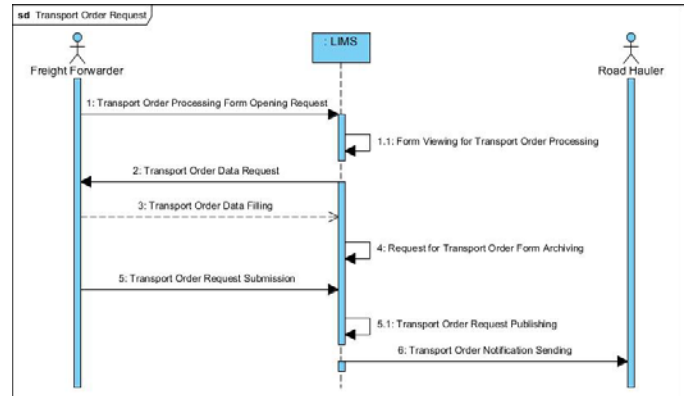


Fig. 6. Sequence Diagram related to Transport Order Management

It may be the case that functional System Requirements written in this way are too complex to be efficiently managed, for example for testability reasons. If this happens, the requirement can be decomposed in smaller and more manageable units, but still maintaining the unity of the main system level requirement.

IV. DEFINITION OF SOFTWARE ARCHITECTURE

The goal of the last phase is to decompose the system into a set of system elements, describe their interactions, and derive from the system level functional Requirements an associated list of requirements to be associated to each system element. The phase is divided in two steps, the first with a purely functional white-box analysis, and the second with the physical analysis, based on the real components chosen as system elements. The approach is recursive, in that each level of system decomposition is requested to perform the same chain of activities, until the lowest significant level for the requirements is reached. Thus, for example, the developer of a system element will receive the Requirements allocated to it, and perform the entire analysis already developed, as if the system level were its customer.

As, for the case under study, the platform is itself a software system, it has been defined through the UML approach [4], [5], [6], [7], [8].

The UML is able to represent the static structure and the dynamic behavior of a software system [2]. A software system is modeled as a collection of discrete objects interacting to perform functionalities that are ultimately exploited by an external user. The UML also contains constructs for organizing models in Packages that enable software teams to divide the software system into smaller parts, identifying dependencies

between Packages, and to manage in this way the installation of these software parts on complex execution environments.

Visual Paradigm is the software tool used to describe the LIMS Platform UML model.





The main UML modeling element is the Component. A component describes a modular piece of a logical or physical system whose externally visible behavior can be described much more concisely than its implementation. It is an abstract element of design that hides its implementation behind a set of interfaces. The Component interfaces describe the functionalities that the component supports. Interfaces may be provided or required. A provided interface describes the operations that a Component guarantees to make available to other components. The component may supply additional operations, but it must at least supply all the operations in a provided interface. A required interface describes the functionality that it needs from other components. The component may not always use all of the listed operations, but it is guaranteed to work if the components that it uses at least supply all the listed operations. Any component that satisfies the set of interfaces can be substituted in a slot. A component is realized by Classes, collected in an "Artifact", which means a file or a set of files, i.e. a physical unit of the construction of a system.

A Subsystem is used to model a wide part of a software system, so Components on large-scale, and it is seen as a <<stereotype>> of a component. A subsystem is therefore a coherent part of the design. It's possible to introduce the concept of module, which is a storage and manipulation software unit. Modules can be associated with a source code, with a binary code and with executables. A stereotype "module" referring to a Subsystem may be defined in Visual Paradigm tool.

The Platform, in general, can be defined as the stereotype "Platform" applied to a Package, which is a part of the Model: this, in fact, is a set of Packages which provides a complete view of the system. The Platform is a set of all modules.

A "Suite" is a collection of modules and represents the Product as sold to a specific Customer. LIMS Platform Modules are shown in Fig. 7. The Modules are classified according to the color codes defined in TABLE I.

TABLE I. SOFTWARE MODULE CLASSIFICATION

Table Column Head		
Module Type	Description	Color Code
Logistic Modules	Modules devoted to the implementation of logistic Processes defined in the Business Model	
Application Modules	Modules responsible for the monitoring of the process status and dataflow management	
SW Core Modules	Modules related to the service software infrastructure	
GIS Module	Module related to the cartographic services	

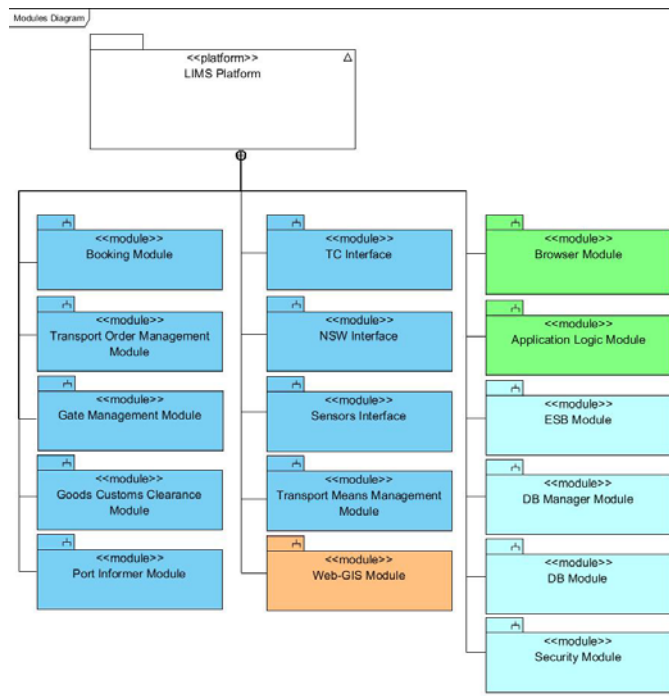


Fig. 7. Software Modules of LIMS Platform

For each Module, a dedicated Component Diagram, showing the UML components defined for it with the related interfaces, has been realized. Fig. 8 shows the Component Diagram for one of the Logistic Modules of the Platform.

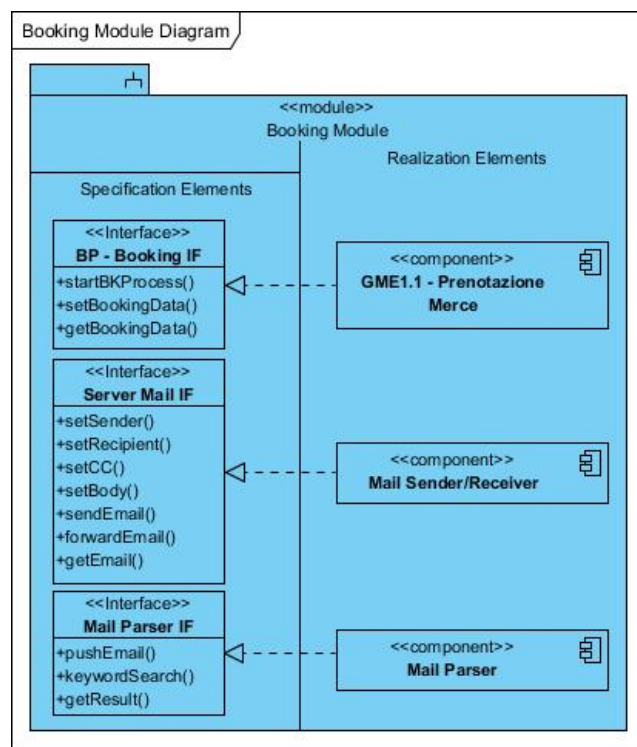


Fig. 8. Component Diagram of a Module

Then, different Suites have been defined as originating from LIMS platform, with the set of modules imported. For each Suite, it has been realized, from one side, a Component Diagram (see for example the diagram in Fig. 9) showing the modules included and the functional relationships between them and, from the other side, a Deployment Diagram (see for example the diagram in Fig. 10) where it is possible to see the physical nodes where the components may be deployed.

After the software architecture modeling, and hence its decomposition in components and related interfaces, the platform software requirements have been derived. A particular attention has been given to the non-functional requirements, which specify requirements under which the system is required to operate or exist or system properties. Quality requirements and human factors are examples of this type. Speaking of product requirements, a focus has been made on reliability requirements, as the LIMS platform has to work in a safe and reliable way. So, the actions of preventive and corrective maintenance, fault types and associated timelines, and methods for instant troubleshooting have been defined. At the same time, the application servers configuration and redundant power supply have been considered. As for privacy requirements, which are of big importance in a platform for the exchange of commercial information, the following requirements related to software security have been defined:

- Confidentiality;
- Integrity;
- Availability;
- Authentication;
- Authorization;
- Logging;
- Platform Session Management;
- Management of errors and exceptions;
- Software configuration parameters Management.

The identification of the platform Modules has been made with this rationale: an ESB (Enterprise Service Bus) is a software architecture with which each application communicates, guarantying the manipulation of the messages and their routing to the final destination. The Application Logic is responsible for the monitoring of the processes status (executed by the Logistic Modules), the presentation of the correct GUI (Graphical User Interface) to the User on the basis of the current status, the management of the data flow in input (from GUI to the DB) and output (from DB or process to the GUI), and the management of the user interactions with the Browser Module. This approach facilitates the requirements allocation and traceability, as the logics of the application is separated from the user interface, and at the same time all the logistic processes, which have to be correctly implemented, are somehow defined with separated software modules.

CONCLUSION

This paper described a model based approach for the derivation of the functional specification and software requirements of an integrated logistic platform which allows an efficient information flows exchange within a port community.

REFERENCES

- [1] S. A. White, and D. Miers, "BPMN Modeling and Reference Guide", 2008.
- [2] J. Rumbaugh, I. Jacobson, and G. Booch, "The Unified Modeling Language Reference Manual", second edition, 2005.
- [3] Technical Board International Council on Systems Engineering (INCOSE). Systems Engineering Handbook, Vrsion 2a, June 2004.
- [4] *UML for Systems Engineering: Request for Information*. <http://www.omg.org/cgi-bin/doc?ad/02-01-17>, 2002.
- [5] *UML for Systems Engineering: Request for Proposal*. <http://www.omg.org/cgi-bin/doc?ad/03-03-41>, 2003.
- [6] *UML 2.1.1 Superstructure Specification*. <http://www.omg.org/cgi-bin/doc?formal/07-02-03>, 2007.
- [7] *UML 2.0 Diagram Interchange*. <http://www.omg.org/cgi-bin/doc?ptc/2003-09-01>, 2003.
- [8] *UML 2.0 Infrastructure Specification*. <http://www.omg.org/cgi-bin/doc?ptc/2003-09-15>, 2003.

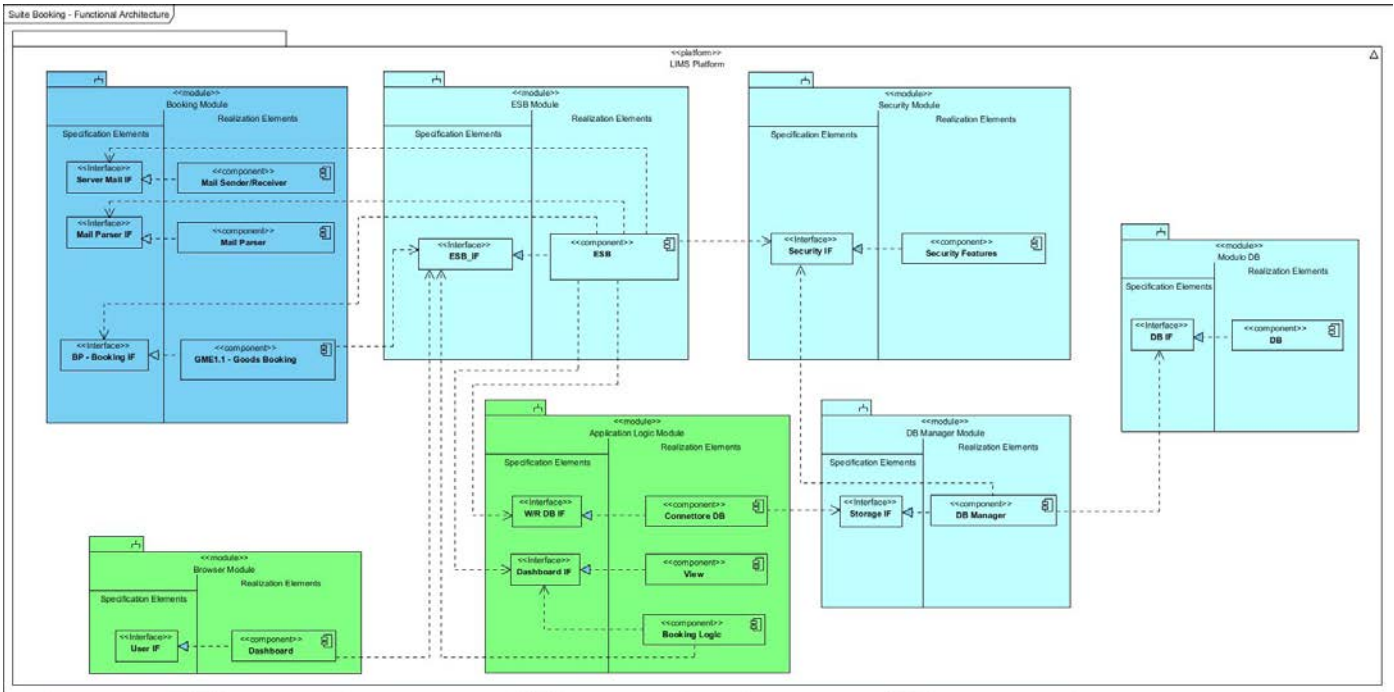


Fig. 9. Component Diagram of a Suite

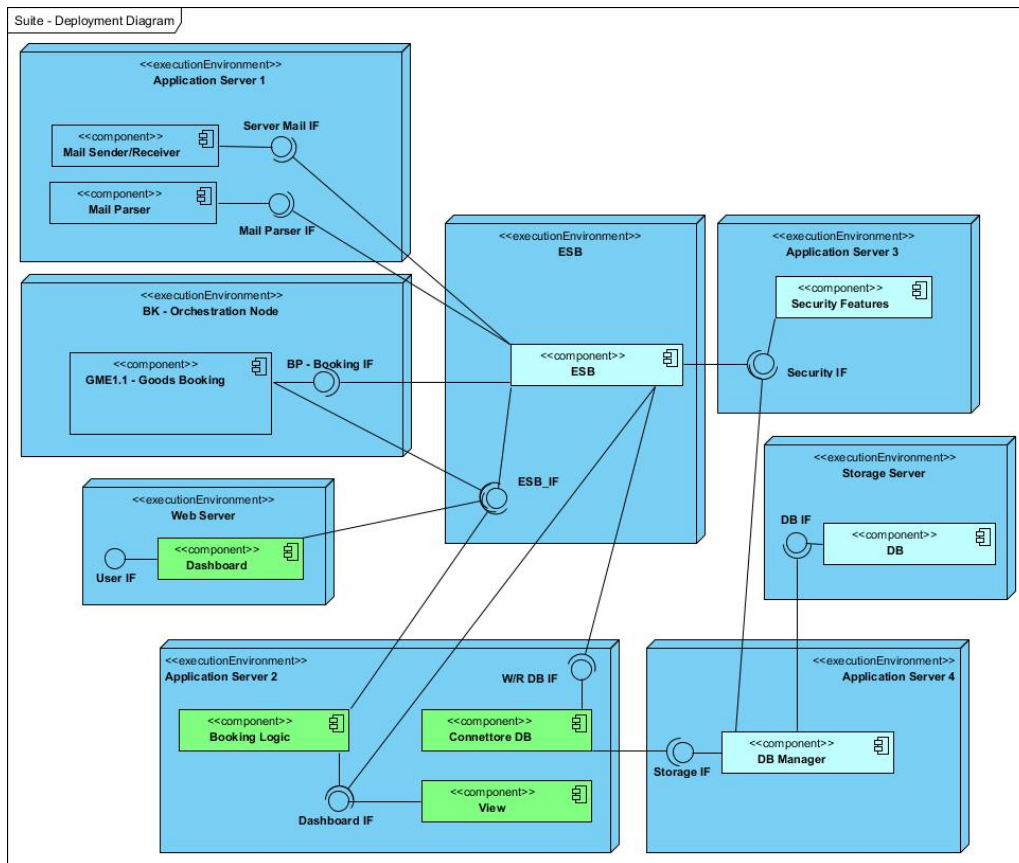


Fig. 10. Deployment Diagram of a Suite