2012 IEEE 6th International Conference ‘Intelligent Systems’

Decision Support Systems Based on a UML Description Approach

Spiros Vasilakos, Giorgio Iacobellis, Chrysostomos D. Stylios
Dept. of Informatics and Telecommunications Technology
Technological Educational Institute of Epirus
Arta, Greece
svasilakos@teleinfom.teiep.gr

Maria Pia Fanti
Dipartimento di Elettrotecnica and Elettronica
Politecnico di Bari
Bari, Italy
fanti@deemail.poliba.it

***Abstract*—Modeling and development approaches for Decision Support Systems (DSSs) attract much attention as technological systems are becoming more complex in order to respond to the constantly growing requirements of nowadays organizational needs, and to support Decision Makers (DM) and managers. In this paper we describe the use of the Unified Modeling Language (UML) to model DSSs. We present a general framework for describing and developing DSSs in a coherent and structured way to all the phases of the development procedure, from the problem definition to the final implementation. Finally, we apply this methodology to a particular case study, the Bay Allocation Problem (BAP).**

Keywords-Decision Support Systems; UML; Modeling; BAP

I. INTRODUCTION

Logistics, business and operational environment nowadays are changing faster than ever before. Organizations must have the capability and flexibility to incorporate, as fast as possible, technological advances, increased supply chain and logistic needs in order to face a constantly growing competition generated by the ever changing consumer habits and needs, the demand for lower production cost, and increased productivity. For many decades, management and decision making was considered an art based on expertise acquired with experience over a long period of time, combined with personal intuition, creativity, and judgment. Although personal qualifications remain valuable, the increasing complexity of modern business environment and the vast volume of available data to be taken into account make the use of advanced modeling and systematic quantitative methods a necessity. A category of computerized systems generally referred as Decision Support Systems (DSSs) are used to address these increased demands. Despite their great variations, almost all DSSs share common characteristics and can be valuable tools for the Decision Maker (DM). For these reasons, their design and development must be done in a precise and structured manner that will reduce the cost of development, both in time and resources, and will produce the desirable results..

In this paper, we present a general framework for modeling and designing DSSs based on the Unified Modeling Language (UML) and the discrete event simulation. To this aim, we present a general structure able to represent a typical DSS. In order to show the efficiency of the proposed methodology, we present a case study facing the Bay Allocation Problem (BAP) in the port of Trieste, a city in the north of Italy. For the sake of clarity, we want to highlight that the aim of the paper is not to address the BAP problem, which, for this reason, is only briefly presented in section 5.

In literature there are a lot of tools able to describe processes or systems, for instance the GRAI method (Graph with Results and Actions Interrelated) [19] or the Business Process Model and Notation (BPMN) [20]. However, UML can be considered a suitable tool to describe a DSS due to its capacity to describe a system from many different points of view and its general purpose nature.

The paper is organized as follows: Section 2 briefly introduces DSSs definition and classifications and describes their advantages; Section 3 describes the suitability of the Unified Modeling Language for modeling; Section 4 introduces a general methodology and framework for modeling DSSs with the use of UML. In Section 5 we apply this methodology to a DSS for the BAP and in Section 6 we mention conclusions and future work.

II. THE DECISION SUPPORT SYSTEM STRUCTURE

There are many different types and approaches for automated decision-making, but most of the times they are computer-based systems used to provide suggestions and advices; they can vary widely according to the activity and the type of problems they support. These kinds of systems are generally called DSSs. Next, three DSSs definitions are quoted:

“Interactive computer-based systems, which help decision makers to utilize data and models to solve unstructured problems” [1].

“Decision Support Systems couple the intellectual resources of individuals with the capabilities of the computer to improve the quality of decisions. It is a computer based support system for management decision makers who deal with semi-structured problems” [2].

“Umbrella term to describe any computerized system that supports decision making in an organization” [3].

As there are various definitions for what a DSS is, accordingly there are various classifications for the different types of DSSs. According to Power [4], DSSs can be divided into five main categories: i) *communication-driven and group DSSs*: in this type emphasis is given to the communication and collaboration of a group of Decision Makers (DM); ii) *data-driven DSSs*: they make extensive use of databases, they process large amounts of data, they use queries and more advanced methods, such as On Line Analytical Processing (OLAP), and they have strong report generation features; iii) *document-driven DSSs*: they are mainly used in knowledge management and retrieval systems, and almost all text-driven DSSs fall in this category; iv) *knowledge-based DSSs*: they make use of Artificial Intelligence (AI) and rules for automated decision making, and are also called expert systems; v) *model-driven DSSs*: models and optimization techniques are used to maximize the desired objectives, DSSs of this type give major emphasis on mathematical models.

According to another classification [5], DSSs can be divided in seven categories: i) file drawer systems, ii) data analysis systems, iii) analysis information systems, iv) accounting models, v) representational models, vi) optimization models, and vii) suggestion models. This classification is related to the influence that the system has on the decisions. Categories i) and ii) are data-oriented, category iii) is data and model- oriented and categories iv), v), vi), and vii) are model-oriented.

Many other classifications of DSSs exist and are based on different characteristics of the systems, e.g. whether they are for personal decision making or group-oriented. If they are based on the type of the application, they can be divided in desktop, web-based or spreadsheet applications, etc. Almost in all classifications, many of the categories can overlap each other or there are hybrids combining two or more categories.

A DSS includes three main components: the data component, the model component, and the interface component [3].

The data component usually consists of a database and a Database Management System (DBMS). The data used can be internal, if they come from internal organization procedures and sources, such as products and services prices, recourse and budget allocation data, payroll cost, cost-per-product, etc. External data can be related with competition market share and government regulations, and may come from various resources, such as market research firm, government agencies, the web, etc. In some cases, a DSS can have its own database or it may use other organizational databases. In this second case, the DSS may directly connect with the databases or may use data available from reports.

The model component mainly includes mathematical models describing the operations of the organization in various levels and the type of functions used according to the operation that they have to support. In some cases, a knowledge management component with functions supporting automated decision making may exist in addition or instead of the model component.

The interface component is the DSS part that is responsible for the communication and interaction of the system with the DM. Its importance, which sometimes may be overlooked, is of high value because, regardless of the quality and quantity of the available data, the precision of the model in describing the organizations procedures and even the hardware capabilities, the interface of the system must ensure that the DM will be able to take advance of the system capabilities.

III. The DSS Model by UML

A. The UML diagrams

The UML is a visual modeling language and has great importance for software engineering. As every language, it has standard notation and syntax. It is composed of thirteen main types of diagrams, each one serving a different purpose and describing the system from different points of view. In particular, we can define two main descriptions [6]:

- **Structural description:** In general, a system is made up of a collection of pieces often referred to as objects. In the UML environment, the system structure is described by diagrams that illustrate the different types of objects from which the system is consisted of and their relationships.
- **Behavioral description:** The behavioral view is very important because the main activity of the system is to ensure that some rules are followed. It is important that these rules are documented in a complete and correct manner, to avoid misunderstanding on both the user and the developer sides.

There are also mechanisms for the extension and combination of these diagrams. UML has been introduced in the domain of software engineering and is widely used since its first release in 1997. However, nowadays, thanks to its nature and the possibility to create customized diagrams, it is adopted as standard modeling tool in a wide kind of applications, from business process to healthcare systems [7].

UML was first developed as a graphical language suitable for software modeling. There are many cases where UML has been used for modeling outside the scope of software engineering. It has been used to model procedures, processes and systems of any kind [8], [9], [10], [11].

The use of UML for modeling has many advantages:

- **Standard notation:** UML has a well-established and documented notation that can be easily adopted to represent a variety of systems. Furthermore, nowadays there are many software tools able to create UML diagrams.
- **Multiple views:** UML diagrams offer multiple views of a system, allowing the representation of different points of view, depending on the phase of study or development. These views can be either static or dynamic, and can represent the structure, the behavior or the types of interactions inside the system.

- **Modularity:** Complex systems consist of many subsystems that may include a variety of activities and procedures. UML allows the modularization of the system, so it is possible to adopt a top-down approach and to use different levels of details for different parts of the system. This feature is very important for the communication among partners, because it helps people with different responsibilities and roles to take care only of the aspect they are interested in.
- **Use cases:** The rights of each actor can be defined with clarity, whether it is external or part of the system, or even another use case.
- **Versatility:** For the representation of a system not all of the diagrams have to be used. Usually, only a subset of the diagrams is used.

B. Modeling a DSS by UML

In this section, the three components of the DSS described in section A are specified.

Data component: UML has been used for modeling databases as well as DBMS [12], [13], [14]; moreover a DSS may use data that are retrieved from external sources, i.e. sensors. With the use of UML, the exact nature and format of the incoming data can be described. Furthermore, hardware parts of a system, such as sensors, can be visualized with the use of application diagrams.

Model component: Many UML diagrams can be used to describe the flow and the control of data to the model component, such as activity, sequence, and state diagrams. The language is used to describe the inputs and outputs of functions. UML has been also used for modeling ontologies and their functionality [15], [16]. Ontologies are one of the main tools in the area of knowledge engineering.

Interface component: User interfaces have been described with UML [17]. They can describe not only attributes and operations of a user interaction with the system, but also different user groups and accordingly variations to the user interface, as it is natural that different users of a DSS will have different needs and rights to the system.

IV. Designing and Modeling a DSS

The modeling of a DSS using a modeling language enables the graphical representation of the system from various aspects, such as the system behavior or its structure. DSSs can become very complex; therefore, their modeling requires to be done in a precise and structured way. Hereinafter, we present a general framework for the procedure of modeling a DSS, the phases and the diagrams used.

In order to design a DSS three main tasks should be satisfied: Communication, Description, and Validation. It is particularly important that the terminology used is clear and comprehensible, in particular to the knowledge carrier (stakeholders, customers, actors involved in the system, etc.), and developers have to understand upon the same requirements. Furthermore, every decision taken should be justified later on. Another important task in the description of the system is that all relevant features of the system need to be presented in a way that is clear to every actor involved into the system modeling. However, there are many difficulties when different description tools are used, for instance different diagrams and notation. It is necessary to overcome this drawback by using standard notation during this phase. For this reason, we have chosen to employ UML, a well-known and widely adopted standard. Despite its diagrams might look a bit complicated at first, when all actors involved get familiar with them, the description task is easily understood. Finally, it is necessary to verify the model in terms of completeness, consistency, and correctness; this task is commonly called validation. Validation is essential in the modeling process because it assures that the behavior of the DSS is as expected to be, when applied to the real system.

In Fig. 1, the stages for designing and implementing a DSS are shown with the use of a UML activity diagram. At the early stages, the knowledge is extracted from the domain experts and data are collected. Next, diagrams that represent the system are made and the suitable model for the DSS is chosen. Finally, the model is built using structural data, and then verified using performance data. At every stage, when inconsistencies are observed, the procedure returns to an earlier stage.

Although DSSs can vary widely, they are generally composed, as already mentioned, by a data, a model and/or a knowledge component and an interface component. In the rest of this section we are going to describe a modeling procedure of a DSS and its components. With the use of this methodology, a DSS can be modeled in such a way that it is, as possible as, independent of the rest of the system. This is particularly important, because this way DSSs can be designed and implemented in already existing systems.

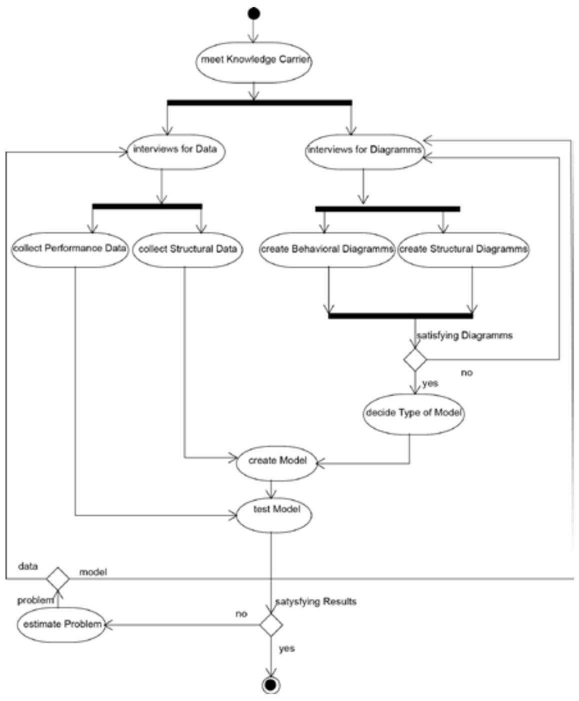

Figure 1. The workflow of modeling a DSS

The first step for building a DSS is the recognition and identification of the problem. This will include the environment where it will be applied, the parameters of the problem, and the expected results. Then, the most suitable model for the specific problem will be chosen.

Next we describe the use of the most appropriate UML diagrams for our purpose.

Use case diagrams: The use of use case diagrams defines the actors of the system as well as how the system is used and interacts with them. Also the boundaries between the DSS and the "outside world", whether it is another system or user of the DSS, are shown.

Class diagrams: With class diagrams the components of a DSS are defined. Each component is represented as a different class. The attributes and operations of every class as well as the types of the attributes and the inputs and outputs of the operations are described. Furthermore, restrictions regarding these values can be shown explicitly. A very important aspect is the use of interfaces. With them, the interactions between the different components of the DSS can be described with clarity; by interfaces the interaction of the DSS with the rest of the system can be shown, allowing this way the implementation of DSS in already existing systems, i.e. ERP's, weather monitoring systems, etc.

Fig. 2 shows the structural description of a DSS with a UML Class diagram. Each component is described with a different class, and its operations and attributes are also described. The communication of the DSS with other actors, such as external DBMS or users, is depicted through the use of interfaces.

Activity and Sequence diagrams: These diagrams allow the description of the behavior of a DSS and all processes composing the system.

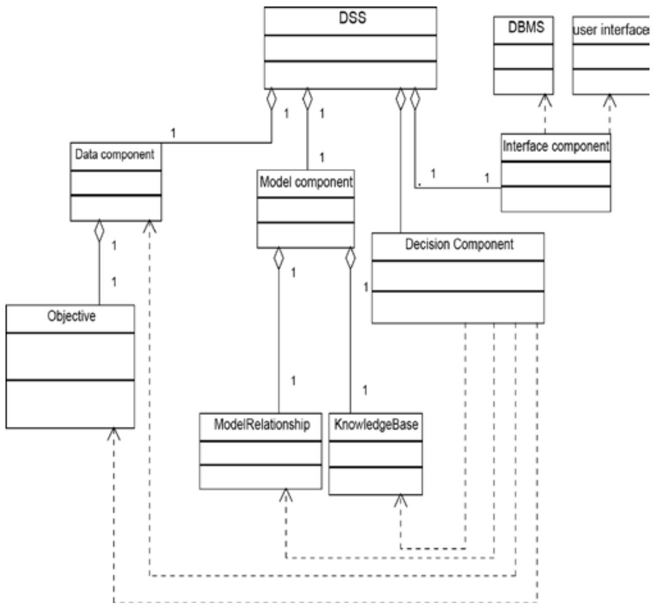

Figure 2. The class diagram of a DSS

Activity diagrams show the workflow of the activities that take place in a DSS and with the use of swimlanes the actors that are responsible for each activity are shown.

The sequence diagrams provide a dynamic description of the system by showing the messages that pass from one component to another or to themselves. The dynamic aspect is added by showing the time flow in the vertical axis. Sequence diagrams allow the description of the flow of information with great detail. Generally, we can say that activity diagrams describe the system from a "choice" point of view and sequence diagrams from a "time-driven" point of view.

Fig. 3 gives a behavioral description of a DSS with the use of a UML Activity Diagram. The use of swimlanes in the activity diagrams allows us to easily show which part of the system is responsible in each phase of an activity. In particular, Fig. 3 concisely describes the structure of the DSS and the corresponding decision procedure.

In this case, the DSS works online and supports DM in real-time proposing decisions and actions in order to reach the desired objectives. The DM inserts a new request in the system by the user interface. Then the Decision component retrieves data about the system from the data component (e.g. state of the system, objectives), analyzes them, and compares the system performance with objectives. Then, it decides whether to evaluate the performances of the system due to a new decision need, if so the model component simulates/calculates the estimated performances based on the model component of the system. Then, if the results are satisfactory, the decision is submitted to the DM by the user interface, otherwise a new decision is evaluated.

V. Case Study Overview of the Berth Allocation Problem

In order to show how it is possible to create a DSS following the approach presented in this paper, we show an example for the Berth Allocation Problem (BAP). We consider a case study involving the Port of Trieste, in the north of Italy.

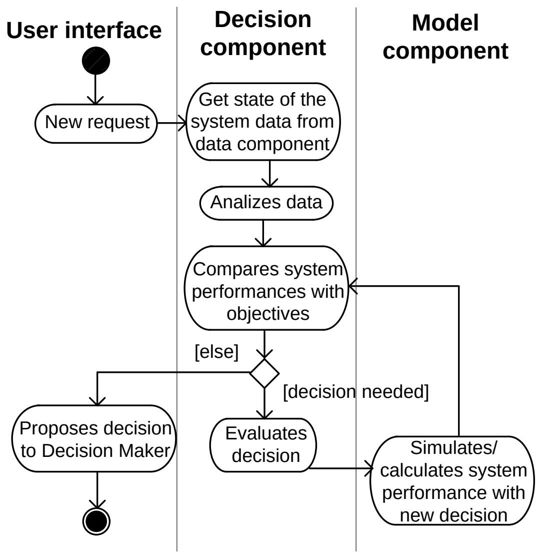
User interface
Decision component
Model component
New request
Get state of the system data from data component
Analizes data
Compares system performances with objectives
[else]
[decision needed]
Proposes decision to Decision Maker
Evaluates decision
Simulates/ calculates system performance with new decision

Figure 3. The activity diagram of a DSS

The BAP consists of assigning incoming vessels to berthing positions. Managers face two interrelated decisions: *where* and *when* the vessels should moor. The problem can be represented in a two-dimensional space (Fig. 4) where vessels are, the vessel length, including a safety margin, and the handling time. These vessels have to be moored without overlapping each other and while satisfying several constraints.

All mooring operations should satisfy constraints relative to the depth of the water (allowable draft) and to the maximum distance in relation to the most favorable location along the quay, computed with respect to the location of the outbound units and to the reserved space for the inbound units. Fig. 4 shows the pick-up and delivery locations m_i, the different typologies of vessel, deep sea vessels (Big) and smaller vessel (feeders). These vessels require different drafts, in this system not all berth position can accommodate a big vessel; furthermore for the sake of generality, in this paper we prefer to talk about units instead of container, because not all containers have the same characteristics. By temporal point of view, all operations should be done within a specific time servicing. Sometimes, time windows are soft and can be relaxed with an appropriate penalty cost.

The handling time of a vessel depends on its berthing point b_i (Fig. 4) and it is a function of the distance from the berth to the pick-up and delivery area of containers stored in the yard. This dependency strongly affects the performance of the port. The handling time data deserves a more detailed discussion. These data depend upon another related decision, which is the number of quay cranes assigned to the incoming vessels, the *Quay Crane Assignment Problem* (QCAP). This decision affects the handling time and has an impact on the BAP. In a complex system, like a transshipment port, the decision making process is often hierarchical, and the QCAP is solved before the BAP. In fact, decisions about the QCAP are subject to less flexibility since the terminal must achieve contractual performance levels, and the number of quay cranes assigned to each vessel is chosen according to practical rules that consider the vessel length and its priority [21].

An estimation of the arrival time of the vessels is known in advance. Every vessel has its own time window determined by its arrival and its maximal allowable completion time. Managers want to minimize both port and user costs, which are related to service time. The objective of the BAP is usually to minimize the total service time of all vessels. Since vessels do not have equal importance, a weighted sum of the vessel service times may better reflect the management practice of some ports. The weights in this sum can represent a pricing scheme or the number of container moves.

Fig. 5 shows the class diagram of the DSS for the BAP. All the classes of the system are described, as well as their attributes and operations. The model component includes the simulation module and the classes that solve the QCAP and the BAP problem. Also, all the vehicles and actors, such as staff members, are described. The interface component describes the communication of the DSS with the DBMS and the users of the DSS.

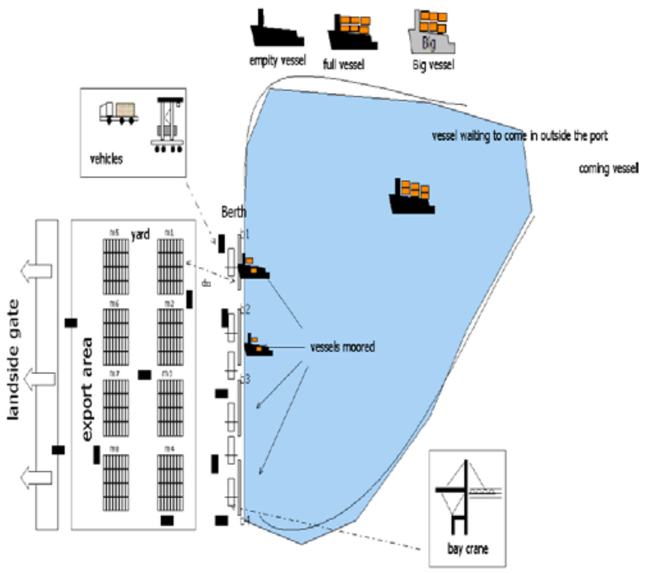

Figure 4. Overview of a bay and the ship birthing

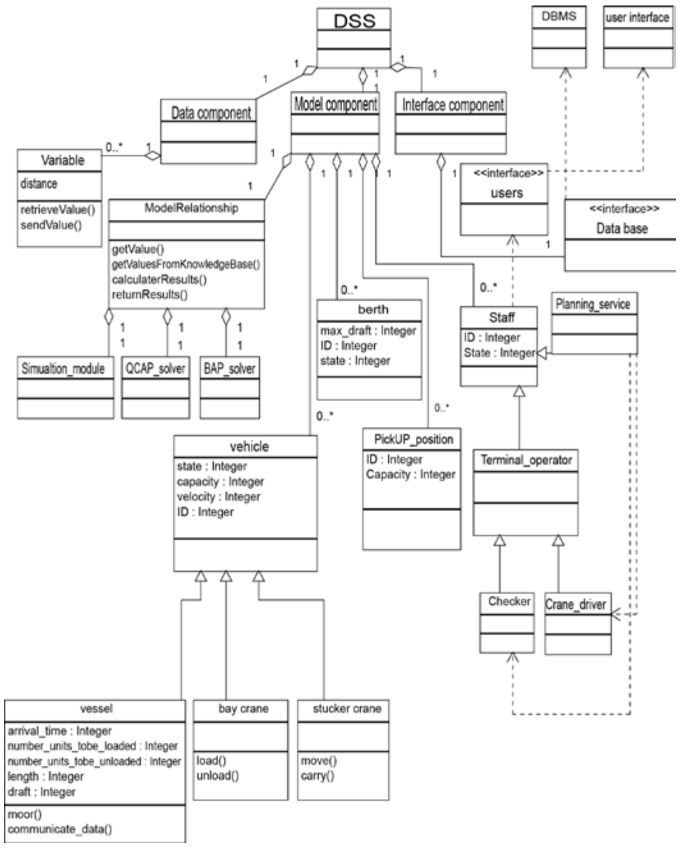

Figure 5. The class diagram for the BAP DSS

Fig. 6 shows how the DSS interacts with the system. When a vessel arrives at the port, the optimization procedure starts and the planning service assigns a berth to the vessel. Hence, the boarding/landing operation can begin.

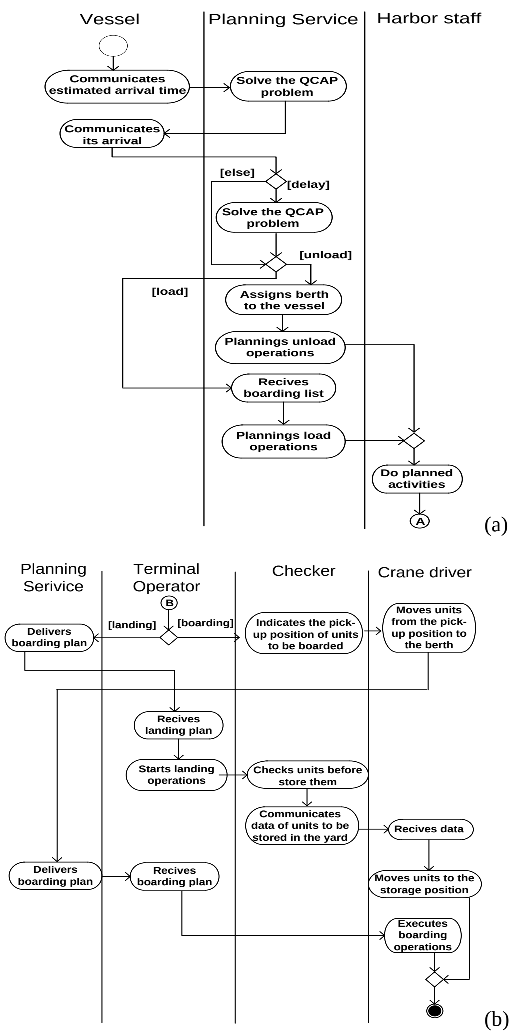
Vessel
Planning Service
Harbor staff
Communicates estimated arrival time
Solve the QCAP problem
Communicates its arrival
[else]
[delay]
Solve the QCAP problem
[unload]
[load]
Assigns berth to the vessel
Plannings unload operations
Recives boarding list
Plannings load operations
Do planned activities
A
(a)
Planning Serivice
Terminal Operator
Checker
Crane driver
B
[landing]
[boarding]
Delivers boarding plan
Indicates the pick-up position of units to be boarded
Moves units from the pick-up position to the berth
Recives landing plan
Starts landing operations
Checks units before store them
Communicates data of units to be stored in the yard
Recives data
Delivers boarding plan
Recives boarding plan
Moves units to the storage position
Executes boarding operations
(b)

Figure 6. (a) and (b) describe the activity diagram for the BAP DSS

VI. CONCLUSIONS

This paper is focused in the utilization of the UML for the description of DSSs to all the phases of their development, from the domain analysis to the final implementation. UML diagrams are used in order to describe a DSS, its components and involved actors and their properties, functions and interactions. Another important aspect of the described approach is that allows DSSs to be integrated in already existing ICT systems in order to support DM. The described case study, in particular the BAP problem for the port of Trieste, illustrates how the described methodology can be applied to specific problems, allowing this way the description and development of DSSs for demanding, real-world problems.

Future work will address the development of DSSs for ICT systems, following the described methodology, especially in the area of logistics and transportations.

ACKNOWLEDGMENT

This work is supported by the E.U. FP7-PEOPLE-IAAP-2009, Grant Agreement No 251589, Acronym: SAIL.

REFERENCES

[1] A.G. Gorry, and M.S. Scott Morton, "A framework for management information systems", Sloan Management Review, vol. 13, pp. 55-70, 1971.

[2] P.G.W. Keen, and M.S. Scott Morton, Decision support systems: An organisational perspective. Reading, MA, USA: Addison-Wesley, 1978.

[3] D. Delen, R. Sharda, and E. Turban, Decision Support and Business Intelligence Systems, 9th Edition. New Jersey, USA: Prentice Hall, 2010.

[4] D. Power, Decision Support Systems: concepts and resources. Westport, USA: Quorum Books, 2002.

[5] S.L.Alter, Decision support systems: current practice and continuing challenges. Reading, MA, USA: Addison-Wesley Publishing Co., 1980

[6] V. Boschian, M. Dotoli, M.P. Fanti, G. Iacobellis, and W. Ukovich, "A metamodeling approach to the management of intermodal transportation networks", IEEE Transactions on Automation Science and Engineering, vol. 8(3), pp. 457-469, 2011, ISSN 1545-5955.

[7] H. Gomaa, Designing Concurrent, Distributed and Real-time Applications with UML. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2000

[8] H.-E. Eriksson, and M. Penker, Business Modelling with UML: Business Patterns at Work. USA: John Wiley & Sons, Inc., 2000

[9] G. Engels, , A. Förster, R. Heckel, and S. Thöne, "Process Modelling using UML", in Process-Aware Information Systems: Bridging People and Software through Process Technology, M. Dumas, W.M.P. van der Aalst, A.H.M. ter Hofstede (ed.). Hoboken, NJ, USA: John Wiley & Sons, Inc., 2005.

[10] S. Kuwamura, A. Matsuda, T. Nakata, M. Shoji, and Q. Zhu, "An object-oriented design process for system-on-chip using UML", Proceedings of the 15th international symposium on System Synthesis (ISSS '02). New York, USA: ACM, 2002, pp. 249-254.

[11] R. Damasevicius, and V. Stuikys, "Application of UML for hardware design based on design process model", Proceedings of the 2004 Asia and South Pacific Design Automation Conference. Piscataway, NJ, USA: IEEE Press, 2004, pp. 244-249.

[12] R.A. Maksimchuk, and E.J. Nailburg, UML for Database Design (1st ed.). Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2001

[13] J.M. Cavero, E. Marcos, and B. Vela, "A methodological approach for object-relational database design using UML", Software and Systems Modeling, vol. 2(1), pp. 59-72, 2003.

[14] W.Y. Mok and D.P. Paper, "On transformations from UML models to object-relational databases", Proceedings of the 34th Annual Hawaii International Conference on System Sciences, vol. 3, pp. 3046, 2001.

[15] S. Cranefield and M. Purvis, "UML as an ontology modelling language", Proceedings of the Workshop on Intelligent Information Integration, 16th International Joint Conference on Artificial Intelligence (IJCAI-99), pp. 46-53, 1999.

[16] X. Wang and C.W. Chan, "Ontology Modeling Using UML", in Proceedings of 7th International Conference on Object Oriented Information Systems Conference (OOIS 2001), Y. Wang, S. Patel, R.H. Johnston (ed.), Calgary Canada, 2001, pp. 59–68.

[17] M. Elkoutbi, R.K. Keller, I. Khriss, and K. Rudolf, "Automated prototyping of user interfaces based on UML scenarios", Automated Software Engineering, vol. 13(1), pp. 5-40, 2006.

[18] J. Araújo, I. Brito, A. Moreira, and A. Rashid, "Aspect-oriented requirements with UML", in M. Kandé, O. Aldawud, G. Booch, and B. Harrison, editors, Workshop on Aspect-Oriented Modeling with UML (UML 2002), Dresden, Germany, 2002.

[19] M. Rahmouni, M.N. Lakhoua, Proposal of the integration of the methods SADT and GRAI in the enterprise, International Symposium on Advanced Engineering & Applied Management, Hunedoara, Romania, 4 - 5 November, 2010.

[20] Business Process Model and Notation, available at: www.bpmn.org/

[21] Cordeau J.-F., Laporte G., Legato P., Moccia L., Models and Tabu search heuristics for the berth allocation problem, Transportation Science, (2005), 39, pp.526–538.