A human-centred design approach for developing dynamic decision support system based on knowledge discovery in databases

Hela Ltifi, Christophe Kolski, Mounir Ben Ayed, Adel M. Alimi

Author version, published in:
Journal of Decision Systems, 22 (2), pp. 69-96, Taylor & Francis, 2013.

A Human-centred Design Approach for Developing Dynamic Decision Support System based on Knowledge Discovery in Databases

Hela Ltifi *, Christophe Kolski **, Mounir Ben Ayed*, Adel M. Alimi*

** REGIM: REsearch Group on Intelligent Machines, University of Sfax, National School of Engineers (ENIS), BP 1173, Sfax, 3038, Tunisia*

hela_ltifi@ieee.org, mounir.benayed@ieee.org, adel.alimi@ieee.org

*** Univ Lille Nord de France, F-59000 Lille, France*
UVHC, LAMIH, F-59313 Valenciennes, France
CNRS, UMR 8201, F-59313 Valenciennes, France

Christophe.Kolski@univ-valenciennes.fr

ABSTRACT. This paper presents a human-centred design approach for developing Decision Support Systems (DSS) based on a Knowledge Discovery in Databases (KDD) process. The KDD process generates a set of software modules. Our approach is based on a critical study of design methods. It uses the Unified Process (UP), which proposes a general framework; however, the UP does not include enough Human-Computer Interaction (HCI) elements. We suggest enriching the UP activities from the HCI perspective, adding HCI elements. The proposed approach is applied to a KDD-based Dynamic Medical DSS.

KEYWORDS: Software process; Dynamic Decision Support Systems; Human-centred Design; Knowledge Discovery in Databases; Human-Computer Interaction

RÉSUMÉ. Cet article présente une approche de conception centrée utilisateur pour le développement d'un système interactif d'aide à la décision (SIAD) basé sur un processus d'extraction de connaissances à partir de données (ECD). Le processus d'ECD aide généralement à générer un ensemble de modules logiciels. Notre approche est basée sur une étude critique des méthodes de conception. Elle utilise le Processus Unifié (PU), qui offre un cadre méthodologique générique. Néanmoins, le PU n'intègre pas assez d'éléments d'Interaction Homme-Machine (IHM). Nous proposons d'enrichir les activités du PU sous l'angle des IHM. L'approche proposée est appliquée à un Système Interactif d'Aide à la Décision Dynamique (SIADD) basé sur l'ECD.

MOTS-CLÉS: Système Interactif d'Aide à la Décision Dynamique, Conception centrée utilisateur, Extraction de connaissances à partir de données, Interaction Homme-Machine.

1. Introduction

1.1. *Dynamic Decision Support Systems based on KDD*

In the real world decisions are taking place in a dynamic environment (Hong et al., 2010). The final decision is only made at the end of some exploratory process such as a KDD process (Peng et al., 2008) (Hong et al., 2010). Dynamic decision making incorporates time constraints. It is characterized by the need to make multiple and interdependent decisions in an environment that changes as a function of the decision maker's actions, environmental events, or both (Brehmer 1992).

Indeed, as it can be seen in Fig. 1, in dynamic decision making, a series of decisions must be made over time based on a set of factors related to the decisional context; the actions are interdependent so that later decisions depend on earlier decisions.

To assist humans in these dynamic decision making scenarios, computer-based decision making systems have been developed. Some examples can be found in management of transportation (Zografos et al., 2002), airline networks (Feigh et al., 2006), healthcare (Lin et al., 2011), customer relationships (Chan et al. 2011), etc. Such systems can be called Dynamic Decision Support Systems (DDSS).

There have been significant improvements in the decision support technologies, related to data storage volume, data processing and data extraction (Shi 2010). We are interested in a new generation of Business Intelligence technology: Knowledge Discovery in Databases (KDD) (Fayyad et al., 1996). This technology can enrich the organization's business intelligence process.

Indeed, traditional decision-support tools (e.g., OLAP[1], Info-center, dashboard, ERP) leave the initiative to the users to choose the elements that they want to observe or analyze. In KDD (Fayyad et al., 1996) (Hand et al., 2001) (Peng et al., 2008), the system often takes the initiative to discover the connections between the data elements. It is then possible, to a certain extent, to predict the future according to the actual new discovered knowledge and those discovered on the past.

Today, decision makers require decision support systems to provide real-time knowledge, especially as the decision is made quickly and dynamically. For example, an airline pilot continuously receives information from his/her dashboard and makes decisions throughout the flight (Cook et al. 2007); an entrepreneur does the same daily to decide the strategic choice for his/her company; a hospital doctor is making measurements and analysis for his/her patients and makes decisions dynamically progressively as the knowledge arrive.

In the field of software engineering several models for specification, design and implementations have been proposed. These models are very limited in the development of human-centred classical decision support systems, thus even more limited for the dynamic systems. In addition the processes dedicated to data mining systems design and evaluation are not covered by the article of (Peng et al., 2008), which is a motivation for this work.

1.2. *Case study used to illustrate the concepts and the approach proposed*

The case study used in the paper to illustrate the concepts and the approach proposed concerns a medical dynamic decision support system aiming at preventing Nosocomial Infections (NI). This kind of infection represents one of the major public health problems. NI are infections contracted in health care institutions. Infections are considered to be NI when they do not exist at the time of the patient's admission (Garner et al., 1988). When the infectious state of the admitted patient is unknown, the infection is classically considered to be nosocomial if it appears 48 hours after the hospitalization.

In the Intensive Care Units (ICU), the NI problem is far more alarming, because the patients who are hospitalized in ICU are more fragile. The importance and complexity of decision-making in controlling NI have been frequently highlighted in the research literature (Brossette et al., 2000). Research has shown the effectiveness of DSS and their capacity to produce useful rules. However, as they are described in the article, the physicians using them appear to have difficulties.

In fact, the decisions on nosocomial infections can reduce the possibilities of acquiring this kind of infection in the ICU, decrease the complexity of patient conditions and clinical interventions; and consequently increase

[1] On Line Analytical Processing

the cost of their care (Sheng, WH et al., 2005). In this context, our system aims at producing daily estimations, in percentages, of the probability to contract a NI during the patient's hospitalization in ICU. This probability is calculated using a data-mining technique and some temporal measures (e.g., urinary catheterization, intubation) in order to predict the infection risks based on patient records (Fig. 1).

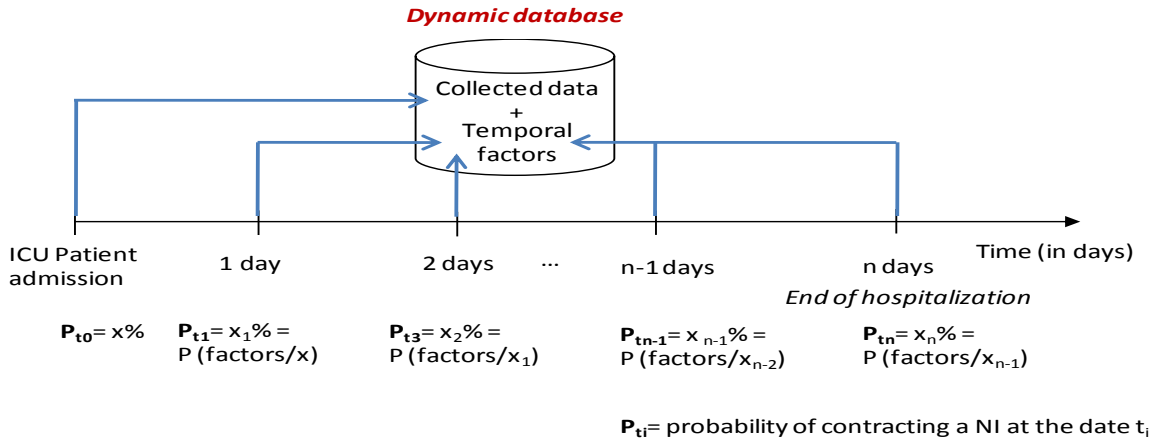

Figure 1. *Dynamic decision making for preventing NI*

Each day, the decision on the patient state depends on the NI probability, and thus on the values of those factors up to the current day but also the previous days, as well as all the knowledge obtained by learning over time and the recording of previous events. In fact, a basic decision is made when the patient is admitted (t0). The future decision is the decision that will be made after the consequences of a basic decision become known. A future decision is linked to the basic decision because the alternatives that will be available in the future depend on the choice made in the basic decision. As time moves on, the future decision at the current decisional stage (t) becomes the basic decision at the next decisional stage (t+1), when new knowledge extracted by data mining (i.e., the probability of acquiring a NI) becomes known, and the future decision(s) should be addressed. As long as the patient is hospitalized, this process of the future decision becoming the basic decision repeats itself. The learn-then-decide-then-learn pattern describes how the decision-maker responds to new knowledge gained during the decision-making process. The elements described above, especially the existence of linked decisions, clearly show that decision-making in NI control is a dynamic process. In this context, the decision-making process requires the consideration over time of linked or interdependent decisions, or decisions that influence each other. This dynamic decision-making pattern is a chain of decisions, interspersed with learning periods. Such system is called Dynamic Medical Decision Support System (DMDSS).

1.3. *Content of the paper*

A DSS based on KDD deals with the decision problem using human knowledge and aims at helping users in the KDD and DSS processes from beginning to end. Human-computer cooperation is essential throughout the decision process, making Human-Computer Interaction (HCI) is a crucial aspect in interactive decision support systems (Piechowiak et al., 2004). The KDD-based DSS design process can be treated as an iterative unified set of activities and operations. We think a development approach relying on the Unified Process (UP) method and the Unified Modelling Language (UML) is appropriate for DSS development. UP tries to build robust system architecture incrementally (Brandas 2007); however, this method does not serve very well when the system studied is highly interactive because UP does not directly and systematically involve the user (Kolski et al. 2001). In this context, it is possible to propose an enriched Software Engineering (SE) development approach from the HCI perspective.

The paper is organized into six sections. In Section 2, we present the theoretical background for this work. In section 3, we propose a human-centred design approach, and its activities are described for each KDD-based DSS modules in a detailed way. In section 4, we introduce our validation process for the new design approach using an example of a DMDSS developed for fighting against NI in the Intensive Care Unit (ICU). In section 5, we discuss our approach, as well as the theoretical and practical implications of this work. Finally, we present our conclusions and our future research perspectives in section 6.

2. Development of Dynamic DSS based on KDD: basic methodological concerns

2.1. *Fundamentals of KDD process*

We consider that the KDD process is iterative and interactive. Iterativity is related to the fact that the KDD is based on a succession of stages and that users can decide to go back constantly if the results are not pertinent for them. Interactivity is related to the various choices that users make. The KDD process has various phases (Fayyad et al., 1996) (Lefébure et al., 2001) (Fig. 2).

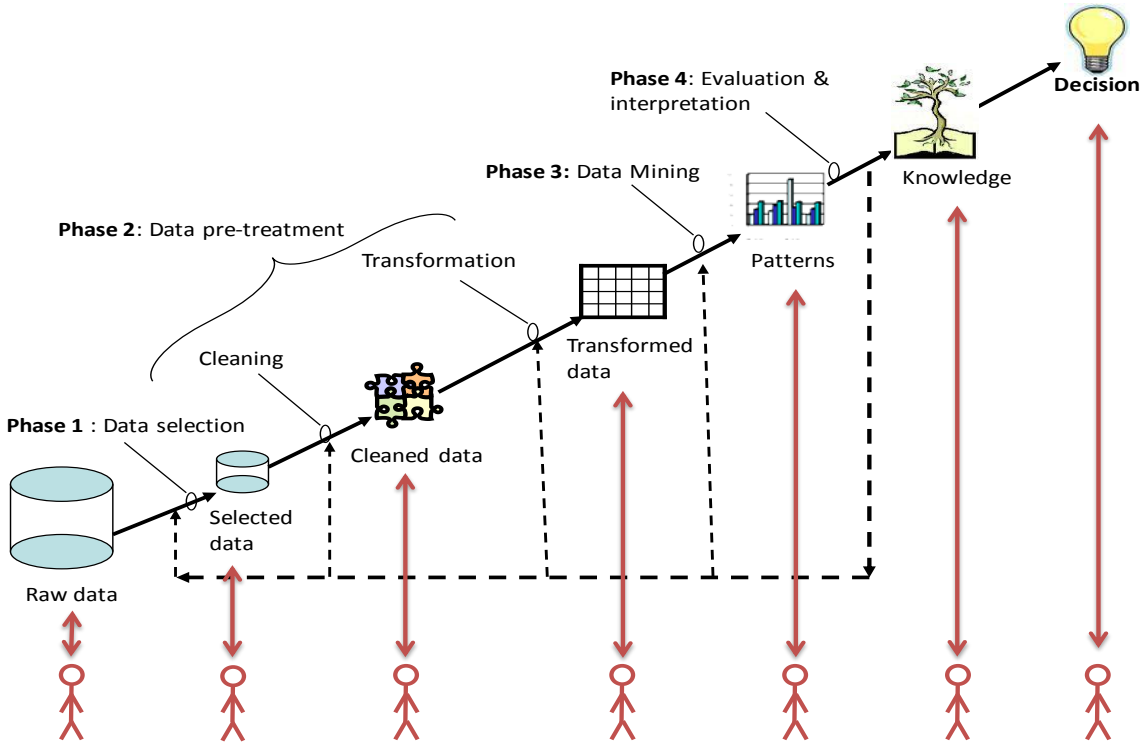

Figure 2. *The KDD process (inspired from Fayyad et al. (Fayyad et al., 1996))*

Different stakeholders are involved at each phase. After defining the objectives and seeking the data needed, the following phases are carried out:

(1) *Data selection* – select the data related to the analysis;

(2) *Data pre-treatment* – clean the data to correct the inaccuracies and errors and then transform the data into a format suitable for data-mining;

(3) *Data mining* – mine the data to extract interesting patterns, by applying one or more techniques (e.g., neural networks, bayesian networks, decision trees);

(4) *evaluation & interpretation* – evaluate and interpret the extracted patterns.

(5) *Knowledge management* – integrate the knowledge in an information processing system (Kanapeckiene et al., 2010).

In the next section, we will briefly examine a set of design models in both the software engineering (SE) and the Human-Computer Interaction (HCI) domains. We will highlight their relevance to the design of KDD-based DDSS, which obviously requires new and adapted approaches.

2.2. *Design models and methods*

In the literature, several analysis and specification processes and methods, intended for the systems Modelling, can be found. Software design models and methods are available for both the SE and HCI domains.

For the SE domain, several development cycles are available, for example, the traditional development cycles, including the waterfall (Royce 2003), V (McDermid et al., 1984) and Spiral (Boehm 1988) models, and the more recent development cycles, including the Y model (André 1994) and the Unified Process (UP) framework (Jacobson et al., 1999). Agile methods also offer promising perspectives (Dyba et al., 2008) (Conboy et al. 2011). All these development cycles aim at producing quality systems. But most traditional development cycles are too often directed towards the technical parts of the system and not towards the user. The more recent development cycles, such as UP, are more directed towards the user, up to a certain level. For this reason, we focus on the UP framework.

The SE models remain the foundation of the methods and models used for human-machine interaction, called HCI-enriched models. Among these models, we can quote Long and Denley's model, which is close to the waterfall model (Long et al., 1990); the Star model (Hix et al., 1993); the Nabla model (Kolski 1997) (Kolski 1998); or the U model (Abed et al., 1991) (Lepreux et al., 2003). The principal concern of all these models is that they highlight the fundamental elements, such as the Modelling of human tasks, the iterative development of prototypes and the evaluation of interactive systems (Kolski et al. 2001). However, they do not necessarily guarantee that a project trying to design and develop an interactive system will be a total success. In fact, no perfect model exists; they all have their strong and weak points.

To propose a development method, we have to take into account two things: 1) the KDD tools are usually difficult to use because most of the users are not experts in computer science or in statistics, and 2) it is difficult to develop a KDD-based DSS that responds exactly to the needs of the users. These difficulties can be overcome by involving the user throughout the KDD-based DSS development cycle.

In this work, we suggest to use the Unified Process, and adapting it to the specificities of DDSM design.

3. Proposed methodology

In the previous section, we have highlighted inherent limits of the well-known development models. For the past few years, the objective of our research is to define a theoretical and methodological framework for designing and evaluating decision support systems, mainly seen as interactive systems (Ltifi et al., 2008) (Ltifi et al., 2009b) (Ltifi et al., 2010b) (Ben Ayed et al., 2010). This framework is based on the Unified Process (Jacobson et al., 1999). Modelling the system architecture with UML, using UP as the main model, helps the developers to rapidly construct and implement accurate, scalable interactive systems (in the sense of (Mohagheghi et al., 2009)). The UP model offers an appropriate framework for developing interactive systems. Since UP is based on UML, developers can create complex decision models and databases (e.g., Data-mining Applications). In addition, UML offers a high level of component reusability.

3.1. *The Unified Process (UP)*

The Unified Process consists of a set of generic principles that can be adapted to specific projects (Jacobson et al., 1999). Thus, to some extent, it is a process pattern that can be adapted to a large category of software systems, various application fields, different company types, different qualification levels and diverse project sizes. UP is (1) controlled by use cases, representing the functional needs of the system; and (2) centred on system architecture, which provides the structure for the work carried out during the iterations. In addition, it is iterative and incremental, with the aim of reducing complexity by controlling it, by breaking up a data-processing project into sub-projects, each representing one iteration. These iterations indicate the steps in the sequence of activities, while the increments correspond to the product development stages. An iteration takes into account a number of use cases. The aspects of the model being analyzed and designed are based on UML (Rumbaugh et al., 1999). A process defines who does what, when and how in order to achieve a preset goal (Jacobson et al., 1999). UP has 4 phases: (1) inception, during which the project scope is defined through use cases and feasibility studies; (2) development, during which the needs are defined and the architecture specified; (3) construction, during which the software is built through several iterations and various system versions; and (4) transition, during which the system is delivered to the end-users and put into service. These end-users are trained and provided with technical support.

Why choose UP as a starting point?

First, UP allows the costs to be limited to the strict expenses of an iteration. It also allows limiting the risks of delaying the installation of the application to be developed. UP also permits potential problems to be identified in the first stages of development, rather than at the testing stage, as it happens with the traditional approaches.

The rhythm of development can be accelerated because the objectives are clear and have been planned in the short term. This short-term planning is due to the fact that the user's needs and the corresponding requirements cannot be completely defined in advance. The system architecture provides the structure for the work carried out during each iteration, while the use cases define the objectives and guide the work completed during the iteration (Larman 2007) (Somé 2006). Second, UP considers the user's needs, as mentioned previously. However, Lemieux and Desmarais (Lemieux et al., 2006) showed that, according to the ISO 13407 standard, UP is not user-centred. This standard specifies the rules to be followed to adapt a software development process to a user-centred design. They also note that the introduction of the use cases is not sufficient to make a design process user-centred.

Our approach must take both SE principles and HCI principles into account. This approach is based on the UP principle of iterative and incremental development, which allows each accomplished task to be evaluated. Our approach thus incorporates the continual and constant participation of the user (Muller 2007).

3.2. *General presentation of the proposed approach*

According to the KDD stages, the system to be developed could require up to five modules for its implementation: (1) select the data, (2) pre-treat the data, (3) mine the data, (4) evaluate and interpret the patterns, and (5) manage the extracted knowledge. The data-mining (DM) module can contain several applications, each one using a different technique to achieve different objectives. Other modules may be combined together; we propose to combine the data selection and pre-treatment modules into a single module called "data acquisition and storage" (Fig. 3).

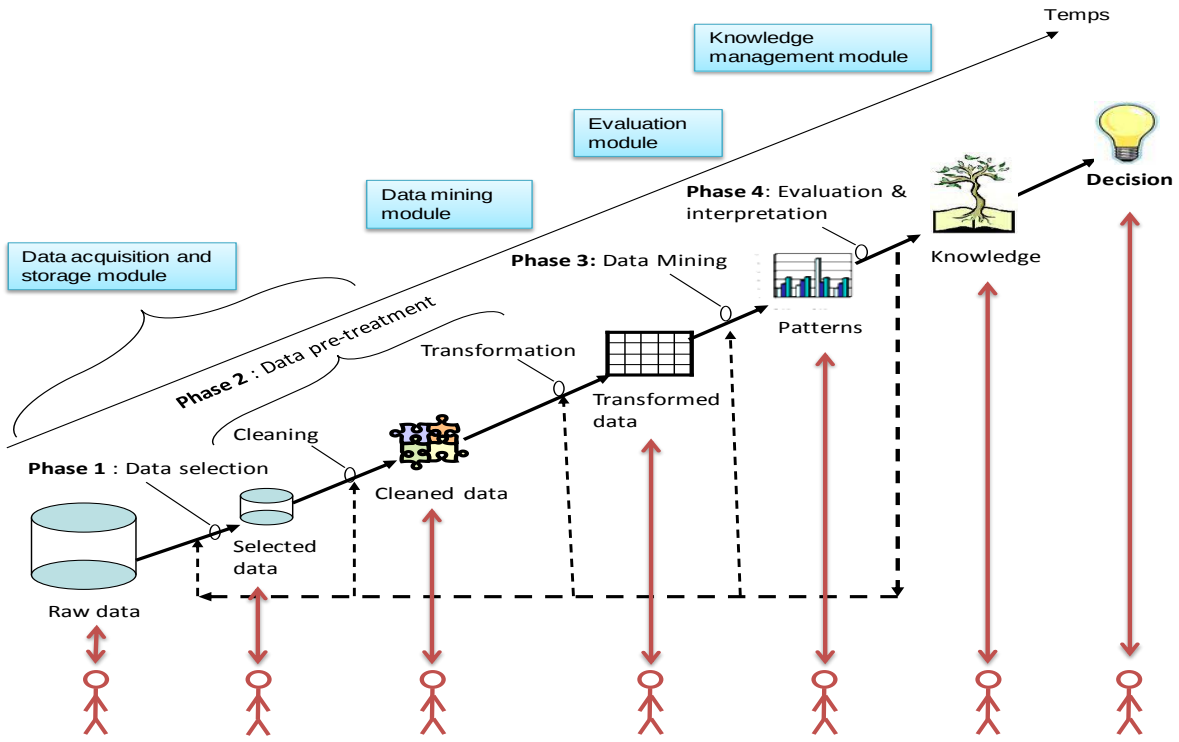

Figure 3. *The KDD-based DSS modules*

We define a methodological framework to design and develop the KDD-based DSS modules. These modules are related since they compose the KDD process. However, each module has its individual objectives. Thus, the design and the creation of each module can be done in parallel or in overlap with the other modules of the KDD-based DSS. Our approach is based on the principles of a user-centred design, as defined by Gould and Lewis (Gould et al., 1985). Several principal phases can be proposed, which are coherent with the different authors who have proposed user-centred methods (Robert 2003). This general development framework is based on the three principles of the generic UP: iterative development, use-case control, and architecture-centred. We propose dividing the project into four phases: Inception, Development, Construction and Transition. The principal

development activities are iterative and incremental. Our approach focuses particularly on the activities, as illustrated by Fig. 4.

For the Modelling, we use UML (Rumbaugh et al., 1999). In fact, UP is based on UML. In addition, it is a language that allows models to be represented without defining their development processes. Thus, it can be used with any other software development process.

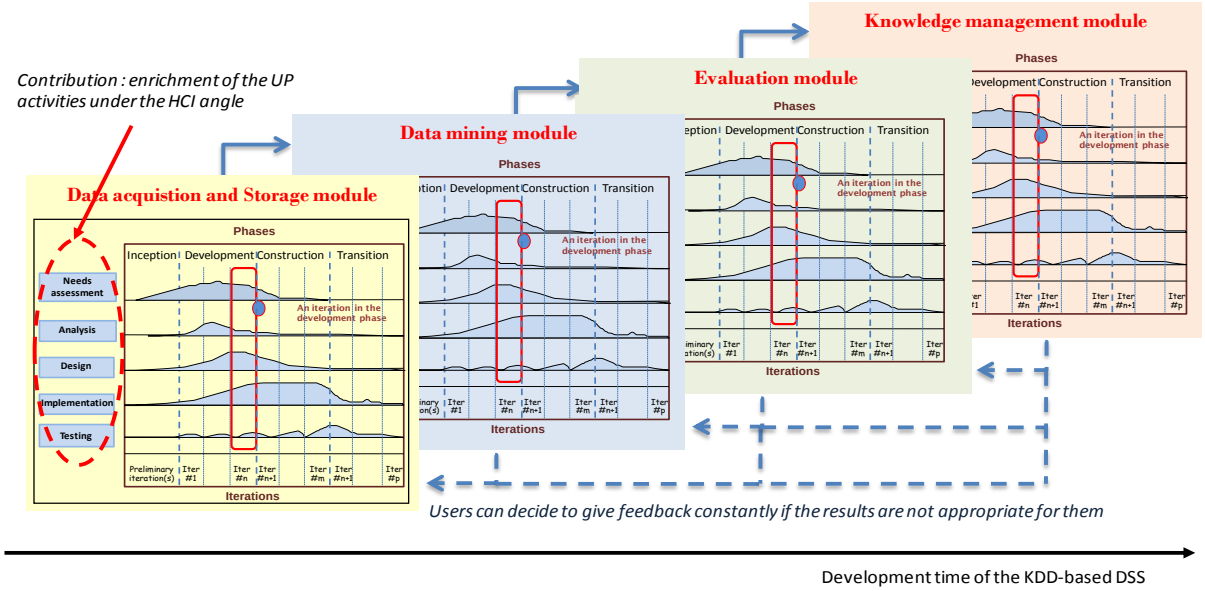

Figure 4. *The proposed approach*

In the following section, we present a detailed description of the HCI-enriched activities used in our approach for developing a KDD-based DSS.

3.3. *Detailed description of the HCI-enriched activities*

For each of the four modules involved in the KDD stages, the following activities have to be performed (see Fig. 5, 6, 7 and 8). Each module has developmental specificities described in Tables 1, 2, 3 and 4.

3.3.1. *The needs assessment activity (Fig. 5)*

Needs assessment consists in analyzing the future decisional system objectives, in terms of the functional and HCI needs. This stage is based on: (1) the analysis of the normal and abnormal decisional situations, which allows the first functional and structural description of the decisional domain; (2) the construction of the first prototypes as soon as possible, in order to involve the future users rapidly by giving them an outline of possible solutions; and (3) the Modelling of the user, which allows the different types of users to be represented, as well as their behaviours. The user model facilitates the design of a user-centred interface. This model is significant in our development approach for interactive systems.

After identifying and expressing the needs, this stage is evaluated early, compared to the system constraints (e.g., organization, logical and/or temporal constraints). This evaluation leads to two possible results: (1) if there is a problem, the user has to give feedback and check the first three stages described above; or (2) if there is no problem, he/she has to identify and organize the tasks that must be accomplished by the Human-Computer pair. These tasks can be divided into three categories: interactive tasks, which involve both the user and the system; automatic tasks, which involve only the application, and manual tasks, which involve only the user. Once the tasks have been defined, they are evaluated.

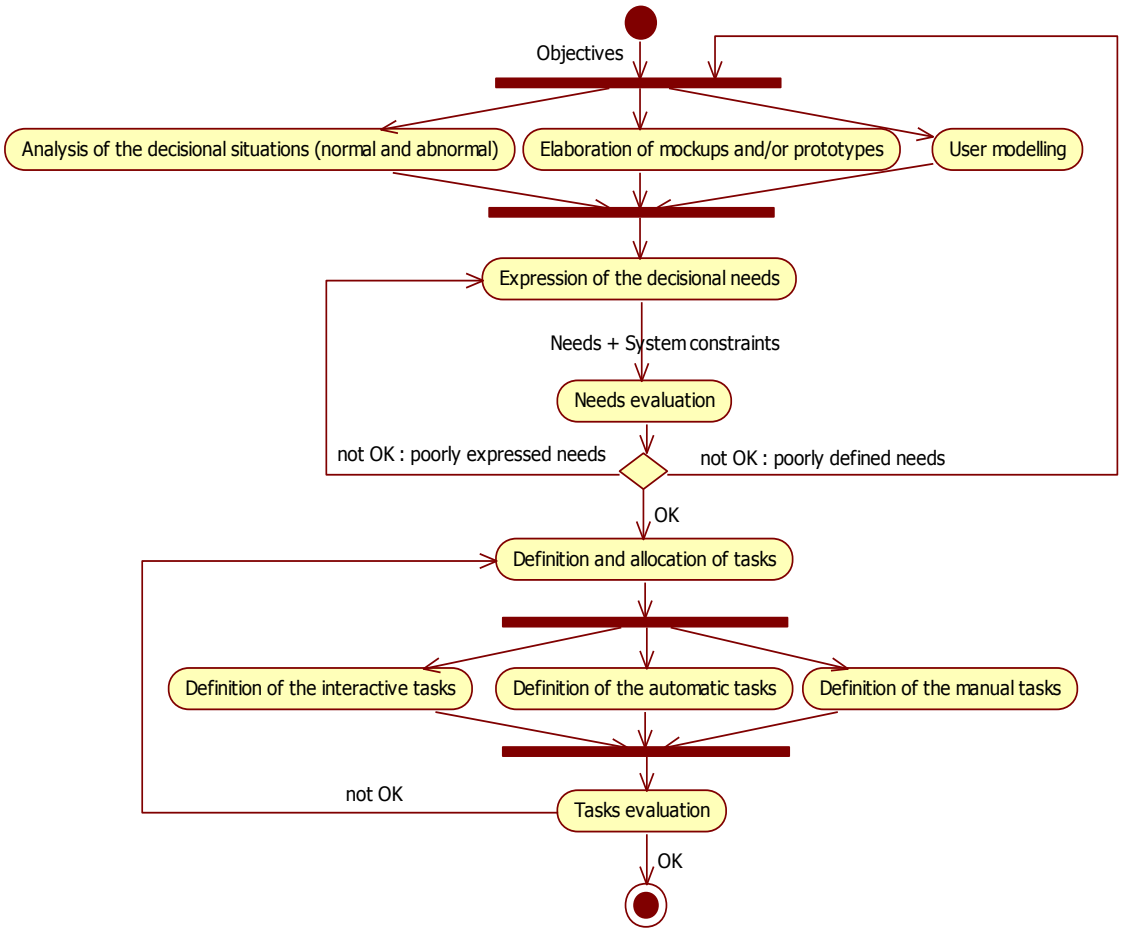

Figure 5. *The needs assessment used for each module of the KDD-based DSS*

For each module of the KDD-based DSS, the needs assessment is done as shown in Table 1[2].

Table 1. *The needs assessment of the KDD-based DSS modules*

Module	Specificities
Data acquisition and storage module	Analyzing the decisional situations from the data acquisition and pre-treatment sub-modules. Preparing the first UI prototypes so that the module general architecture can be determined. Modelling the user, who is an expert in his or her field and in using data-processing tools. Defining the sub-modules' functionalities of data capture, selection, cleaning, transformation and demonstrating them to the users for an early evaluation. Defining and describing: (1) interactive tasks (e.g., choosing the data acquisition zones, the data to be selected and cleaned, the variables to be transformed), (2) automatic tasks (e.g., the automatic operations of pre-treatment) and (3) manual tasks (e.g., users filling files with the necessary decisional data). Evaluating tasks in order to check whether or not they satisfy the user's needs.
Data-mining module	Analyzing decisional situations related to the various data-mining techniques that can be used. Building the UI prototypes that present (1) the way that the user wants to visualize extracted knowledge, and (2) the variables that have to be introduced by the user. Modelling the user, who must be able to interact with the module but he is not supposed to know the details of the data-mining technique(s) to be used. Evaluating the data-mining functionalities to check the possibilities of applying the selected data-mining technique(s). Dividing the tasks of this module into interactive tasks (e.g., introducing the necessary values for the automatic tasks) and automatic tasks (e.g., executing the selected data-mining techniques). Evaluating the possibilities offered by these techniques.
Evaluation module	Studying the evaluation criteria of the discovered models (or patterns), as well as the resulting decisional situations. Creating the interpretation methods for the patterns of the knowledge extracted. Preparing the UI prototypes to give an idea about the interface architecture to evaluate and interpret the patterns. Modelling the user, who must be able to interpret the results of the knowledge discovery and execute a useful and usable evaluation module (Bahloul et al., 2010) Validating the evaluation and interpretation functionalities. discerning the task evaluations: (1) interactive tasks for a qualitative evaluation; (2) automatic tasks for a quantitative evaluation Interpreting automatically in order to extract knowledge.
Knowledge management module	Describing each decisional situation in order to represent a value of knowledge and a possible solution used to rectify this situation. Displaying the UI prototypes that present (1) the values predicted by data-mining techniques and (2) the possible solutions automatically generated by the system. Modelling the user, who takes on the behavior of a decision-maker. Validating the functionalities of prediction, possible solution generation, and decision proposal, based on the extracted knowledge. Dividing the tasks into: (1) interactive tasks, which allow the decision-maker to validate or cancel previously-generated solutions, and (2) automatic tasks, to predict the values and generate possible solutions. Checking whether or not tasks allow the decision-maker to make satisfactory decisions.

[2] Several stakeholders can be involved in each module: the user (i.e., decision-maker), the KDD expert, the designer and the human factors specialist.

3.3.2. *The analysis and design activities (Fig. 6)*

The first activity results in a list of interactive and automatic tasks, as well as the user model. Once the tasks have been defined, they must be analyzed. The stage of interactive task analysis specifies the human tasks in normal and abnormal situations (Abed et al., 1991), which is closely connected with identifying the user's characteristics, resources and cognitive limits in the user model. Analyzing the automatic tasks is related to the analysis of the functional tasks that can be accomplished within a decision-making process. The User Interfaces (UI) can then be analyzed in order to define their behaviour. This analysis focuses on the relationships between the user and the interactive system. It is a question of indentifying the ergonomic and technical needs rigorously, then defining the number of screens to be used, the sequence of views, the information presentation modes and the human-computer dialogue methods.

This specification stage is followed by the design stage, which allows the models for the automatic tasks and the UI to be designed. This design makes it possible to formalize the needs specification to define the appropriate algorithms. These algorithms will be transmitted to the implementation stage to be developed. The analysis and design models are evaluated at this level. This evaluation checks that they correspond to the needs. If the result is unsatisfactory, the analysis and design models are modified to produce models that correspond to the needs.

Once the specification and the UI design have been validated, an advanced UI prototype can be built in collaboration with the experts and the users. All the information provided within the framework of these activities makes it possible to specify the architecture of the future system. This system is developed and enriched progressively by adding the analysis and design models, which are UML models that integrate the sequence, communication and interaction overview diagrams. The system architecture is composed of the following elements: (1) the user interface allowing interactions with the user; (2) a database containing information, for example, about the users, the procedures, the decision problems and the solutions; and (3) the software packages. This architecture must be evaluated. Then, depending on the evaluation's results, either we start the following activity or we go back to an earlier point and make modifications.

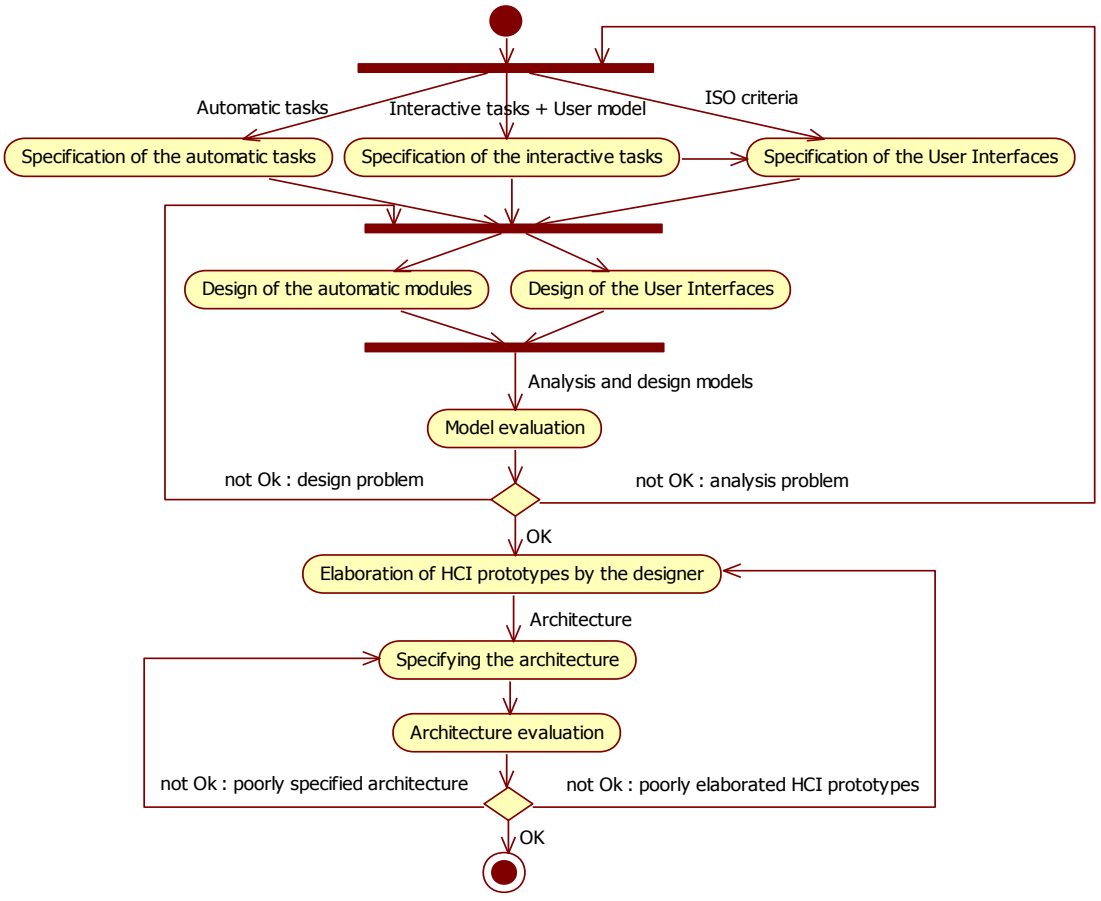

Figure 6. *The analysis and design activities used for each module of the KDD-based DSS*

For each module of the KDD-based DSS, the analysis and the design activities are done as shown in Table 2[3].

Table 2. *The analysis and design of the KDD-based DSS modules*

Module	Specificities
Data acquisition and storage module	Analyzing the interactive and automatic tasks for the data capture, selection, cleaning and transformation sub-modules. Formalizing the UI and the associated functionalities for data acquisition and data pre-treatment. Designing the way a new recording will be filtered in the database, and cleaned and transformed if necessary. Evaluating the data capture mechanism and the pre-treatment of the analysis and design models. Preparing the advanced prototypes for data selection, cleaning and transformation. Evaluating the data capture mechanism and the pre-treatment of the sub-module architecture. The architecture includes: (1) the data capture and pre-treatment UI, (2) their software packages, and (3) the database.
Data-mining module	Analyzing the automatic tasks allows the specification of the data-mining execution process and the user-machine interactions. Formalizing the various data-mining sub-modules. Evaluating the analysis and design models to check whether or not the data-mining techniques are formalized correctly. Giving a good idea of the possible user-module interactions with the advanced prototypes. Evaluating the data-mining module's architecture. The architecture includes: (1) data-mining application UI, (2) data-mining algorithm(s), and (3) the database.
Evaluation module	Analyzing the automatic pattern evaluation and interpretation tasks and the interactive evaluation tasks. Defining the design formalisms of the quantitative and qualitative evaluation as well as the interpretation for the knowledge extraction. Validating the design models concerning the different patterns evaluation and interpretation methods. Preparing the advanced UI prototypes presenting the way that the user will be able to interact with the module to evaluate the patterns. Evaluating the evaluation and interpretation of the sub-modules' architecture. The architecture includes: (1) the evaluation UI, (2) the interpretation UI, (3) the evaluation and interpretation tools, and (4) the database.
Knowledge management module	Analyzing the prediction and the solutions generation tasks. Analyzing the interactions between the decision maker and the system (to validate or not each decision generated by the system). Designing the various tasks of prediction and possible solutions generation for the decision support in their various possible scenarios. Validating (1) the analysis and design formalisms, (2) the UI and (3) the knowledge management for decision-making algorithms. Showing how the user will be able to validate or cancel a possible automatically-generated solution through the advanced UI prototypes. Evaluating the knowledge management module's architecture. The architecture includes: (1) automatic prediction UI, (2) UI for the automatic generation of possible solutions, (2) automatic prediction and solution-generation algorithms, and (4) the database.

[3] Several stakeholders can be involved in each module: the user, the KDD expert, the designer and the human factors specialist.

3.3.3. *The implementation activity (Fig. 7)*

The Implementation starts with a schedule for the software development and maintenance. This schedule must be prepared to facilitate the long-term feasibility of this development effort. The implementation activity consists in coding the functional parts, based on the previously-defined algorithms (in the design activity), and the UI. All the code components are then assembled and integrated (Prencipe et al., 2005) in a subsystem in order to build a prototype at the end of the iteration.

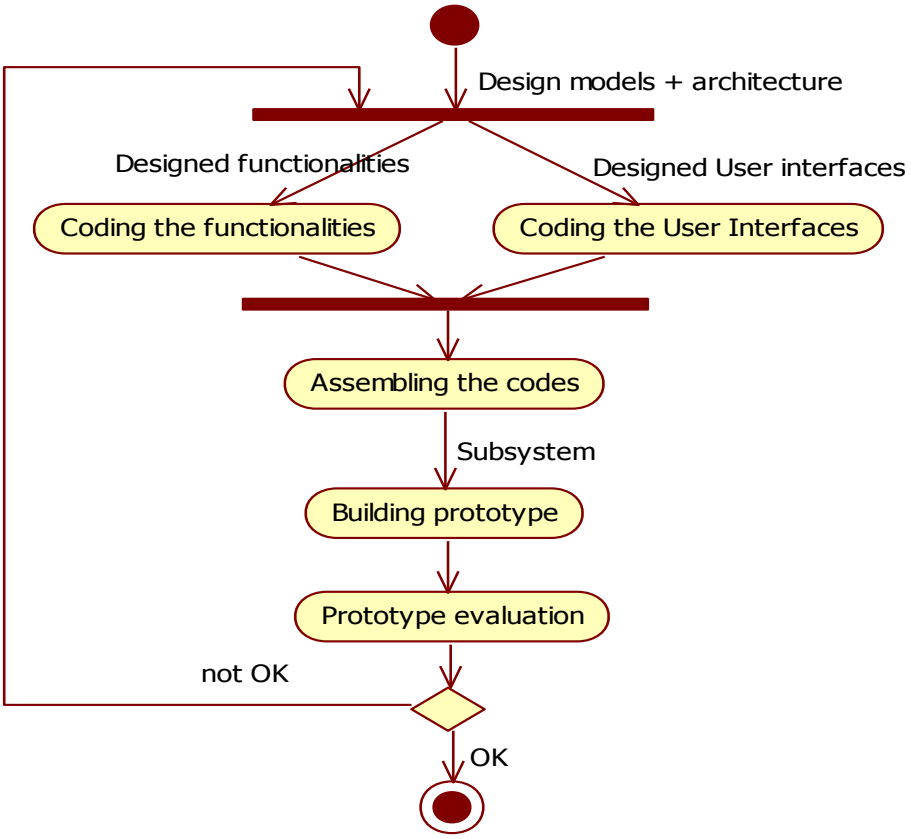

Figure 7. *The implementation activity used for each module of the KDD-based DSS*

The prototypes represent the levels of future software development that take into account more and more details of the specifications. In particular, effectiveness, robustness and maintenance are taken into account from the start of the development, since prototyping encourages data abstraction. Each prototype is evaluated in order to verify its correspondence to the system architecture. Each prototype must be validated by the user and the KDD expert to make sure it conforms to technical and user-friendly standards.

For each module of the KDD-based DSS, the implementation is done as shown in Table 3[4].

[4] Several stakeholders can be involved in each module: the user, the KDD expert and the developer.

Table 3. *The implementation of the KDD-based DSS modules*

Module	Specificities
Data acquisition and storage module	Implementing (1) the data acquisition and pre-treatment UI, (2) the database, and (3) the software packages. Assembling the code components for the UI and the software packages in order to build the prototype. Evaluating the prototype by the user (data acquisition tasks) and by the KDD expert (data pre-treatment tasks).
Data-mining module	Coding the data-mining UI and the data-mining algorithms. Assembling the code components for the UI and the software packages for the data-mining techniques in order to build the prototype. Evaluating the prototype by the KDD expert.
Evaluation module	Coding the evaluation and interpretation UI and software packages. Assembling the code components to build a prototype. Evaluating the prototype by the user (qualitative and quantitative evaluation tasks) and by the KDD expert (interpretation tasks).
Knowledge management module	Implementing the UI and the software packages for the prediction, possible solution generation and decision-making sub-modules. Assembling the code components to build a prototype. Evaluating the prototype by the user (i.e., the decision-maker).

3.3.4. *The testing activity*

The testing activity makes sure that the users are able to execute their tasks through the proposed UI. This test will highlight the errors in the code. The detected errors can be functional, connected to choice and performance, or interactive. We distinguish two kinds of tests: unitary tests and integration tests (Fig. 8)

Unitary tests evaluate the functions that were developed during the iteration. Testing can begin as soon as the function is coded, checked and, if necessary, validated. When non-conformity is detected, it is then necessary to correct the errors and the anomalies. If no non-conformity is detected, the code components will be gradually assembled with the code tested in the preceding iterations. The UI of the subsystems must also be tested, as well as the way each one of them communicates and behaves in the new environment. This evaluation generally focuses on the performance of the total system, according to user behavior when they are interacting with the system. Once the real model of the resulting subsystem has been established, it is compared with the ideal model defined in the needs assessment. This comparison allows the subsystem to be validated in terms of the defined needs (Abed et al., 1991). The utility and the usability of the system are also evaluated.

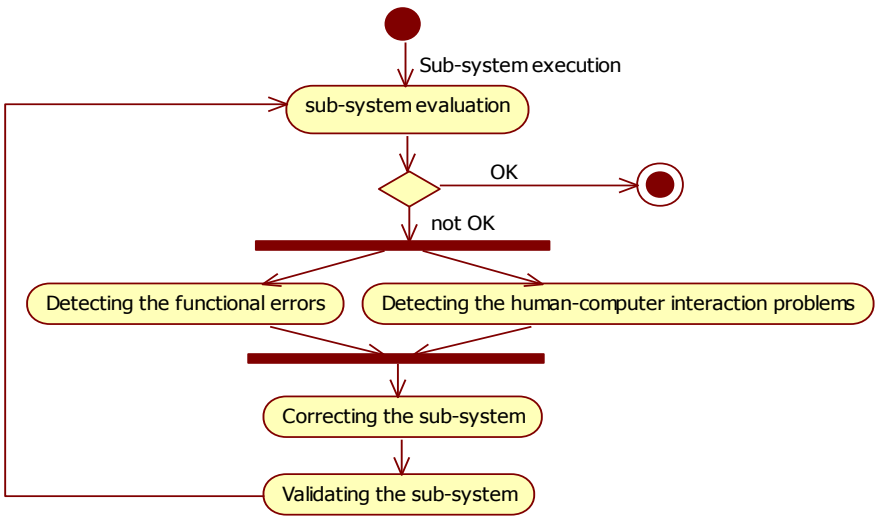

Figure 8. *The tests used for each module of the KDD-based DSS*

Integration tests evaluate the complete system. They check the interactions between the subsystems in order to make it possible for the users to construct their decision-making process. This evaluation must insure that the system provides a complete assistance tool and verify the quality of the UI.

For each module of the KDD-based DSS, the testing is done as shown in Table 4[5].

Table 4. *The testing of the KDD-based DSS modules*

Module	Specificities
Data acquisition and storage module	Executing the user tests to check if the user can carry out data capture and pre-treatment operations using the UI proposed. Executing the integration tests to validate the first KDD-based DSS module and to check the user-system interactions during the data capture and pre-treatment operations.
Data-mining module	Executing the unitary tests to check if the UI are useful for: Verifying whether or not the data-mining algorithm satisfies user needs, and verifying the handling ease of the UI. Testing the complete data-mining module to make sure that the user can use it easily.
Evaluation module	Executing the unitary tests to check whether or not the user manages to deal with the evaluation tasks. Executing other tests by the KDD expert to verify whether or not the discovered patterns can be interpreted in order to extract knowledge to support the decision. Executing the integration tests to make sure that: starting from all of discovered patterns, the system will be able to evaluate and interpret them in order to extract knowledge.
Knowledge management module	Executing the unitary tests to check whether or not the users can interact easily with the generated solutions and make decisions. Testing the UI of this module by the users to validate them. Executing the integration tests to validate the decision-maker's interactions with the computer during the process.

Our approach proceeds with several iterations from the inception phase to the transition phase. We applied the different activities in each iteration, along with their actions (described above). The resulting models are represented using UML. In the next section, we present the application of our approach in the healthcare domain.

4. Case study

In the preceding sections, we presented our global approach for developing an interactive KDD-based DSS. This approach was applied to a concrete case in the healthcare domain in order to help physicians to understand and prevent NI. The system is currently being used in the ICU of the Habib Bourguiba Teaching Hospital in Sfax, Tunisia (Ltifi et al., 2010a). This KDD-based Dynamic Medical DSS was designed and developed according to the four phases of the U P. Each stage of these iterations used the activities suggested in this paper. These activities and their actions were not conducted with the same intensity in each iteration.

Let us take the example of user Modelling. This Modelling process began gradually at the beginning of the project and became dominant in the first iteration of the development phase, but it was reduced to almost nothing in the last iterations of the construction and transition phases. This process shows the need to proceed step by step in close association with the users. They progressively suggest new improvements and tests in view of the results obtained in the preceding iterations.

The development of the KDD-based DMDSS was subdivided in four modules: (1) data acquisition and storage module; (2) data-mining module; (3) evaluation module; and (4) knowledge management module (Fig. 3). At the time that this article was written, our user-centred, iterative, incremental process had resulted in (1) the creation of a temporal database; (2) the development of a data acquisition and storage module; (3) the

[5] Several stakeholders can be involved in each module: the user, the KDD expert, the developer and the human factors specialist.

development of a data-mining module, with two data-mining applications; and (4) the development of a knowledge management module. The evaluation module is now under development (Bahloul et al., 2010).

The three first modules were designed and implemented according to the tables presented in the section 3.3. Table 5 shows the development of the data acquisition and storage module.

Table 5. *The design process for the data acquisition and storage module*

Phase	Iteration	Activity	Brief description
Inception	Iteration 1	Needs assessment	- Nosocomial Infections (e.g., definition, causes, risks) were studied. - The general architecture of UI prototypes was outlined, with the physicians proposing windows with tabs. - A preliminary user model was created by an expert in the NI field. - Initial use case model (diagram + textual description) was created, including initial data acquisition and data pre-treatment functions. - Tasks were defined as presented in section 3.3.
		Analysis & design	*Non-applicable (N/A) in this iteration*
		Implementation	*N/A in this iteration*
		Testing	*N/A in this iteration*
Development	Iteration 1	Needs assessment	- Filling out the forms created an additional workload to select the data. To compensate for this increased workload, the users asked us to offer interns full database tables to reduce their workload by allowing them to avoid writing table names.
		Analysis & design	- The automatic data acquisition modules (e.g., age, duration of stay, risk of death and data selection) were analyzed using collaborative activity and state/transition diagrams and designed using sequence diagrams. - The interactive data acquisition modules were analyzed and designed.
		Implementation	- The raw database was implementation using the SQL Server DBMS.
		Testing	*N/A in this iteration*
	Iteration 2	Needs assessment	- In the data pre-treatment UI prototypes was added the possibility of an automatic search for a data inaccuracies or errors in the database. - The thorough analysis of NI allowed us to refine the definition of the tasks to be executed by the decision-maker;
		Analysis & design	- The automatic data selection modules and HC interfaces were analyzed and designed. - The automatic data cleaning modules and HC interfaces were analyzed and designed.
		Implementation	- Only the temporal database was implemented.
		Testing	*N/A in this iteration*
Construction	Iteration 1	Needs assessment	- Since the database is temporal, it is useless to record the date each time a physician enters the system; so a button for "Next Day" and "Previous Day" would facilitate the data entry. - The UI was modified by verifying the boxes or radio buttons to facilitate the data entry in order to avoid errors and to make the data coherent, especially the values on which certain statistical tests are based.
		Analysis & design	- The automatic data transformation modules and the UI were analyzed and designed.
		Implementation	- The various data acquisition UI were implemented,

Phase	Iteration	Activity	Brief description
			taking all the users remarks into account. - The automatic data acquisition modules were implemented, using C#.net
		Testing	- A proposal was made to add explanatory icons associated to some buttons (e.g., +, Stop).
	Iteration 2	Needs assessment	*N/A in this iteration*
		Analysis & design	*N/A in this iteration*
		Implementation	- Some slight improvements to the data acquisition UI (i.e., text displays) were made. - The automatic data selection modules and the UI were implemented. - The automatic data cleaning modules and UI were implemented. - The automatic data transformation modules and UI were implemented.
		Testing	- The user tests were executed. Results: the usability problems related to data entry were described, and the other physician needs were detected.
Transition	Iteration 1	Needs assessment	*N/A in this iteration*
		Analysis & design	*N/A in this iteration*
		Implementation	- The last bugs connected to the latest user tests were corrected.
		Testing	- The last prototype was tested using the ICU patient data transcribed on the provided forms, and no errors were detected. - The main user expressed his satisfaction and accepted the final module "data acquisition and storage ".

According to the needs expressed by the users, we developed two applications for the data-mining module in order to predict every day the NI appearance in ICU. Each application uses one data-mining technique. Table 6 shows the development of the data-mining module.

Table 6. *The design process for the data-mining module*

Phase	Iteration	Activity	Brief description
Inception	Iteration 1	Needs assessment	- The physicians understood the methodological principles. They found it very useful to be involved. - For this module, the original data-mining technique used was the K Nearest Neighbours (KNN). - The users prepared several UI prototypes that showed, if the probability is higher than 50%, there is a NI risk. - The use case model included the KNN technique functionalities. - The tasks were defined: • Automatic tasks – The distances between the new patient characteristics and those of the patients already hospitalized, recorded in the ICU database were calculated. We used the Euclidian distance. The other automatic task is the identification and classification of the K nearest neighbours. Based on this classification, we calculated the probability of contracting a NI. • Interactive tasks – with the goal being to have the best discrimination rate, the user chose the value of K (1, 3 or 5), thus following up the classification process.
		Analysis & design	*N/A in this iteration*
		Implementation	*N/A in this iteration*
		Testing	*N/A in this iteration*

Phase	Iteration	Activity	Brief description
Development	Iteration 1	Needs assessment	- The project team considered the KNN technique to be insufficient for dynamic decision-making, since the data were temporal and multidimensional (Fu 2011). For this reason, we added the Dynamic Bayesian Networks (DBN) (Darwich 2001) (Murphy 2002) as an analysis technique to obtain knowledge models that evolve over time. This technique is a special Bayesian Network, which is used for the dynamic stochastic process models. - The use case diagram was refined to add the DBN technique function: the daily classification of the patient state, which evolves throughout the patient's hospitalization.
		Analysis & design	- The Euclidian distance calculation and the associated UI were analyzed and designed.
		Implementation	- The database is ready to mine.
		Testing	*N/A in this iteration*
	Iteration 2	Needs assessment	- A "Report" button was added to launch a procedure recapitulating the patient's hospitalization history.
		Analysis & design	- The necessary procedures were added to display the patient data corresponding to the selection criteria.
		Implementation	- The first UI and the automatic module for calculating the Euclidian distance were implemented.
		Testing	*N/A in this iteration*
Construction	Iteration 1	Needs assessment	*N/A in this iteration*
		Analysis & design	- The diagrams were modified to reflect the additional tasks. - The rest of the automatic and interactive tasks were analyzed and designed.
		Implementation	- The second UI was implemented, showing the probability of an NI appearance (Fig. 10).
		Testing	*N/A in this iteration*
	Iteration 2	Needs assessment	*N/A in this iteration*
		Analysis & design	- The analysis and the design of the DBN technique's function must take into account that patient data are treated progressively. This data enrich the DBN models that use them to provide other predictions. - With each time series (Fu 2011), it is possible to predict the patient state. Then, this prediction is used as an entry for predicting the patient state on the following day. In addition, the current day observations are also entries for predicting the NI contracting result.
		Implementation	- The UI and the automatic module for dynamic prediction of the patient state using the DBN technique was implemented (Fig. 10).
		Testing	- User tests were executed that compare the specified tasks and the tasks really carried out by the user, with globally successful results. Some remaining usability problems were highlighted. - The transactions were tested using SQL Server DBMS
Transition	Iteration 1	Needs assessment	*N/A in this iteration*
		Analysis & design	*N/A in this iteration*
		Implementation	*N/A in this iteration*
		Testing	- The users expressed their satisfaction and accepted the final data-mining module.

Fig. 9 shows the User Interface for using the KNN algorithm. This algorithm is run each day during the patient's hospitalization. In this article, we cite the example of a patient for which the probability of contracting an NI is modified: it is 0% on the first day and 60% on the fifth day. Such a prediction helps the physicians to take the necessary precautions to protect the patients during their ICU stay.

As shown in Fig. 10, after choosing the patient file number (485/06), the physicians[6] (i.e., the system user, who is an expert in this domain) only have to add the current date for which they want to obtain the NI probability for the patient. For the classification, a column named "results" is reserved and is initialized at "not".

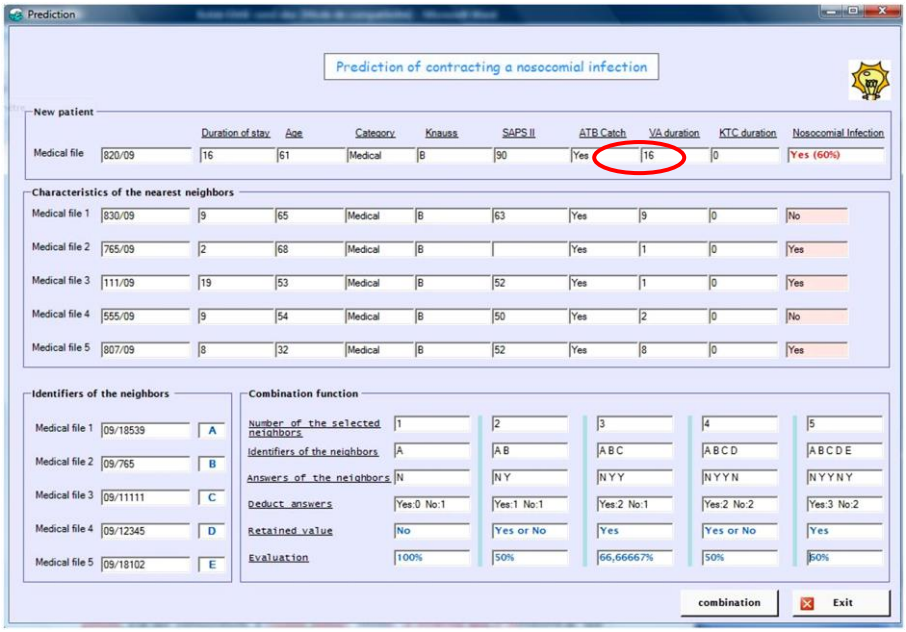

Figure 9. *Human-Computer Interface for NI prediction*

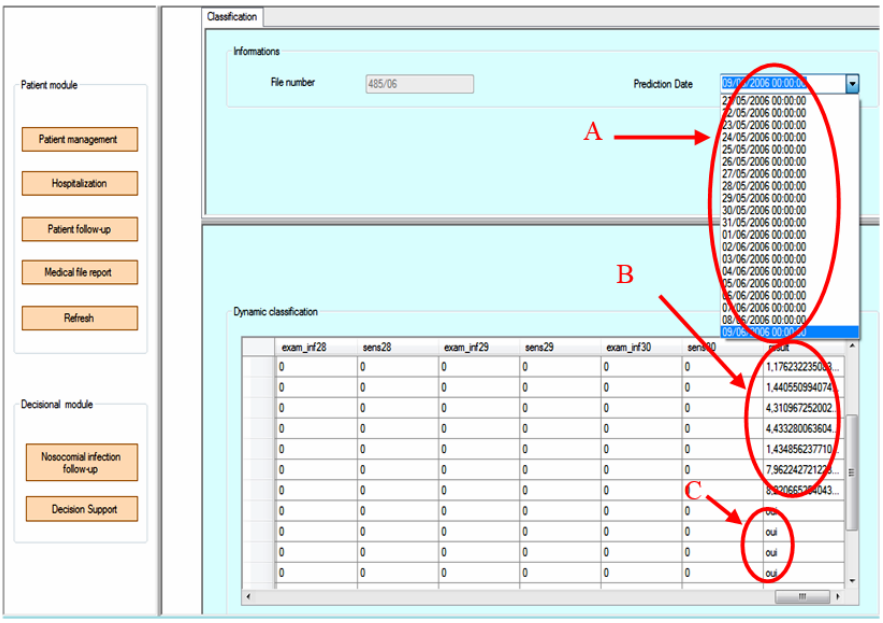

Figure 10. *Human-Computer Interface of dynamic classification of the state of the patients*

For each date and each patient selected, a daily probability of contracting NI will be associated to the patient. If, on one date, a "yes" value is recorded, then this patient will be classified having a nosocomial infection starting at this date. If a "no" value is recorded, the physician can predict the varied probabilities during this time period if the patient will contract a nosocomial infection. In Fig. 10, there are three ovals: (1) the first, denoted A, indicates the identifiers of the time series associated to prediction dates; (2) the second, denoted B, shows the various probability values of contracting an NI from the first days of the patient's hospitalization, with the last

[6] They are the principal stakeholders for this project.

value on this list of probabilities being 82%; and the third, denoted C, shows whether or not the patient has contacted an NI after this period of time. Once the data-mining module was implemented, we started developing the Knowledge management module. The objective of this module was to choose between various possible alternatives to solve the NI appearance problem by selecting the best solution or a compromise. Table 7 shows the development of the Knowledge management module.

Table 7. *The design process for the knowledge management module*

Phase	Iteration	Activity	Brief description
Inception	Iteration 1	Needs assessment	- According to the probability value discovered by the data-mining technique(s), a list of possible solutions must be generated. Fig. 11 shows a user's handwritten note with suggestions for a UI for the possible solutions generated according to the probability of contracting a NI.
		Analysis & design	*N/A in this iteration*
		Implementation	*N/A in this iteration*
		Testing	*N/A in this iteration*
Development	Iteration 1	Needs assessment	- The physician proposed associating a text and an alarm image if there is a great risk to the NI occurrence percentage.
		Analysis & design	- The functions of knowledge integration and automatic generation of possible solutions were analyzed and designed.
		Implementation	*N/A in this iteration*
		Testing	*N/A in this iteration*
	Iteration 2	Needs assessment	- Tasks were added: typing a name or an administrative code results in a display of the patients concerned by the decision; double-clicking on the line corresponding to patient results in a display of their contact information.
		Analysis & design	- The UML diagrams were modified to reflect the additional tasks. - The automatic generation of possible solutions function was analyzed and designed.
		Implementation	- The knowledge integration function was implemented.
		Testing	- The UI showing the NI occurrence probability was tested.
Construction	Iteration 1	Needs assessment	*N/A in this iteration*
		Analysis & design	*N/A in this iteration*
		Implementation	- The automatic generation of possible solutions function was implemented.
		Testing	- The generation of the possible solutions to prevent NI was tested, resulting in the detection of some errors in the transactions with the temporal database. These errors were corrected.
	Iteration 2	Needs assessment	*N/A in this iteration*
		Analysis & design	*N/A in this iteration*
		Implementation	- The decision-making function was implemented. - Whenever possible, the data entry zones were replaced by radio buttons or check boxes.
		Testing	- User tests were executed. The results met our expectation; the tasks were accomplished, though with some UI overlap.
Transition	Iteration 1	Needs assessment	*N/A in this iteration*
		Analysis & design	*N/A in this iteration*
		Implementation	*N/A in this iteration*
		Testing	- User tests related to the dates were executed, resulting in a simplification of the data entry procedures. - User tests were executed to verify the proposed UI had the characteristics of a quality interface (e.g., coherence, error prevention) and validate the module. - The users expressed their satisfaction and accepted the final "knowledge management module".

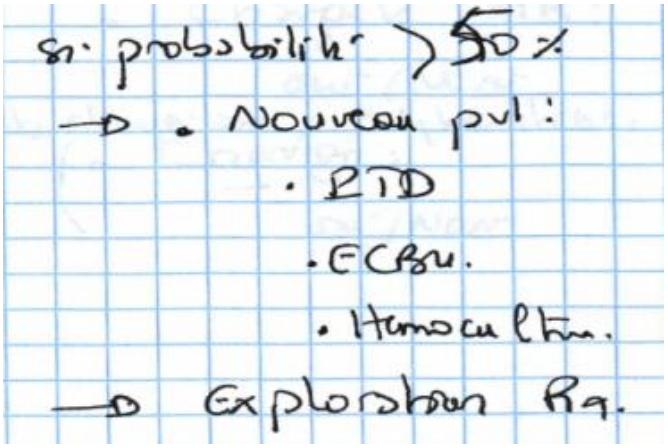

Figure 11. *Extract of the possible solutions generation UI (as initially expressed by a physician)*

5. Discussion

5.1. *Methodological contributions*

In this paper, we showed that KDD is a process that helps decision-makers to obtain knowledge that supports the best decision. KDD and DSS converge because DSS can be based on KDD process. A KDD-based DSS can be considered as an interactive system, composed by a predefined set of modules. The process starts by defining the problem and the objectives and ends by extracting knowledge to help make decisions. As far as we know, no one has proposed any human-centred approach to design and developed a KDD-based DSS.

In this work we propose an approach in which the user is the principal actor all through the development period. Therefore, our approach can be a model that guides the developer to build the system, while respecting interactivity and interactivity. In the literature, we can find many models for SE development or enriched from the HCI perspective. But none of these models is appropriate for developing a KDD-based DSS. By studying the SE model, we found that UP is the most appropriate process for interactivity and iterativity. However, as UP is not user-centred, we enriched it with HCI elements. The HCI elements include: elaboration and design of mock-ups and prototypes, user modeling, human task modeling, early evaluations, UI specification and design.

The most important research contribution of this paper is to propose a generic approach that helps to develop a KDD-based DSS by involving the end user all through the development process in order to make it possible for potential users to describe their functional needs and to evaluate and validate the different interfaces.

5.2. *Practical contributions*

Our practical contribution is the development of a KDD-based Dynamic Medical Decision Support System (DMDSS) to fight against nosocomial infections. This DMDSS reduce their risk and their impact across the continuum of care. Health care organizations are required to do use such DSS.

During this project, we were able to involve physicians and KDD expert throughout the development process. The final product is set up in the intensive care unit (ICU) of Habib Bourguiba hospital in Sfax - Tunisia (Ltifi et al., 2010a). Our KDD-based DMDSS operates in that unit, and it is used to explain and prevent nosocomial infections. For the system performance evaluation, we used a test database that contains 58 cases (i.e., patient files). We applied two data-mining algorithms – KNN (Riesbeck et al., 1989) and DBN (Darwich 2001) (Murphy 2002) (Trabelsi et al., 2010) – to real data from the ICU. We were able to extract knowledge and transform it automatically to obtain probabilistic, quantitative and qualitative prediction results. These prediction results are 74% reliable, which is very promising.

Our KDD-based DMDSS predicts the patient state. This prediction is dynamic, evolving throughout the patient's hospitalization through new measurements. These measurements enrich the models in order to help provide other predictions. At each dayi of the patient's hospitalization, his or her state is envisioned at some

future point by a probability, which will be used to predict dayi+1. We compared these predictions with observed results. We obtained the results presented in the confusion matrix[7] in Table 8.

Table 8. *The confusion matrix of the results produced by the DBN*

Observed results / Predicted results	Yes	No	Total
Yes	9	8	17
No	7	34	41
Total	16	42	58

We calculated the rates of evaluation starting from the prediction results obtained by our DBN structure (Ltifi et al., 2010a). We found that the classification rate was correct to 0.74, the positive capacity of prediction = 0.56, and the negative capacity of prediction = 0.81. The observed vs. predicted results presented in Table 5 are represented by the histogram in Fig. 12.

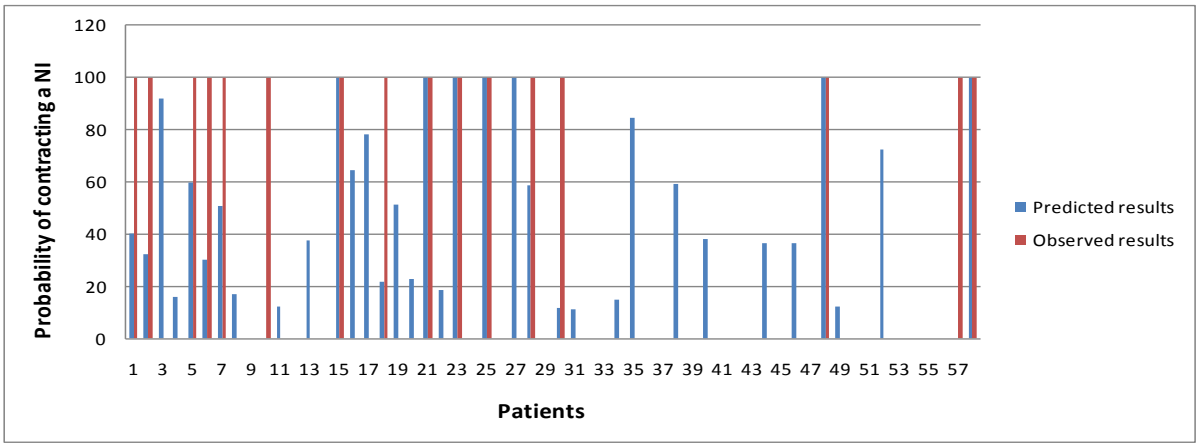

Figure 12. *Prediction results for a database of 58 patients (Ltifi et al., 2010a)*

6. Conclusion

This article proposes a user-centred approach for designing DSS based on KDD. The DSS deals with the problem according to its knowledge. Some of this knowledge can be extracted using a decisional support tool, called Knowledge Discovery in Databases (KDD). Since KDD-based DSS are highly interactive, designers of such systems must rely on elements from two fields: Software Engineering (SE) and Human-Computer Interaction (HCI).

In this context, we have proposed an approach that uses UP and UML, to which we contributed the enrichment of each activity with HCI elements. Developers must have a unified view of system design and implementation. We recommend using the proposed approach to develop an integrated DMDSS based on the activity workflows, which will be able to accurately identify, model and implement the decision-making requirements. This approach should allow a realistic, well-documented system, which will be able to meet most user requirements and support the development of an accurate and flexible DSS.

In order to validate this approach, we are presently developing a KDD-based DMDSS to supervise the NI contracted in ICU. The project is being developed in constant association with the system users (i.e., physicians): needs assessment, task definition, user characteristics identification, prototyping, evaluations and validations. The DSS is not totally completed. Further studies will be conducted to improve our approach. We also want to propose a methodology for evaluating KDD-based DMDSS, taking into account evaluation methods and techniques used in two fields: HCI and visual data-mining (Ltifi et al., 2009a). The integration, early in the

[7] Yes: the patient has a NI
No: the patient does not have a NI
Total: the total predictions

project, of conceptual models considering explicitly the context would constitute another research way (Brossard et al., 2011).

7. Acknowledgements

The first, third and fourth authors would like to acknowledge the financial support for this research by grants from the ARUB program under the jurisdiction of the General Direction of Scientific Research (DGRST) (Tunisia). Thanks also to all the ICU staff of Habib Bourguiba Teaching Hospital for their interest in the project and all the time they spent helping us design, use and evaluate our system. The second author would like to thank the Nord-Pas de Calais region, the French national government and the FEDER program for their financial support.

8. References

Abed, M., Bernard, J.M., and Angué J.C. (1991), 'Task analysis and modelization by using SADT and Petri Networks', *Proc. Tenth European Annual Conference on Human Decision Making and Manual Control*, Liege, Belgium.

André, J. (1994), *Moving From Merise to Schlaer and Mellor*, SIG-Publication 3.

Bahloul, R., Ben Ayed, M., Alimi, M.A. (2010), 'Vers une méthode d'évaluation de Système Interactif d'Aide à la Décision basé sur le processus d'ECD', in: Atelier : *Évaluation des méthodes d'Extraction de Connaissances dans les Données, 10ème Conférence Internationale Francophone sur l'Extraction et la Gestion des Connaissances*, Hammamet, Tunisia.

Ben Ayed, M., Ltifi, H., Kolski, C., and Alimi, M.A. (2010), 'A User-centred Approach for the Design and Implementation of KDD-based DSS: A case Study in the Healthcare Domain', *Decision Support Systems*, 50 (1), 64-78.

Brehmer, B. (1992), 'Dynamic decision making: Human control of complex systems', *Acta Psychologica*, 81 (3), 211–241.

Brandas, C. (2007), 'Unified Approach in the DSS Development Process', *Economy Informatics Review*, 11 (1), 98-102.

Brossard, A., Abed, M., and Kolski, C.(2011), 'Taking context into account in conceptual models using a Model Driven Engineering approach', *Information and Software Technology*, 53, 1349–1369.

Brossette, S.E., Sprague, A.P., Jones, W.T., and Moser, S.A. (2000), 'A data mining system for infection control surveillance', *Methods of Information in Medicine*, 39, 303-310.

Boehm, B. (1988), 'A Spiral Model of Software Development and Enhancement', *Computer*, 21, 61-72.

Campanella, G., and Ribeiro, R.A. (2011), 'A framework for dynamic multiple-criteria decision making', *Decision Support Systems*, 52, 52–60.

Chan, S.L., and Ip W.H. (2011), 'A dynamic decision support system to predict the value of customer for new product development', *Decision Support Systems*, 52, 178–188.

Conboy, K., and Morgan, L. (2011), 'Beyond the customer: opening the agile system development process', *Information and Software Technology*, 53 (5), 535-542.

Cook, M. (Eds), Noyes, J. (Eds), Masakowski, Y. and (Eds) (2007), *Decision-making in Complex Environments*, London, Ashgate, 424.

Darwich, A. (2001), 'Constant-space reasoning in dynamic Bayesian networks', *International journal of approximate reasoning*, 26, 161-178.

Dyba, T., and Dingsoyr, T. (2008), 'Empirical studies of agile software development: A systematic review', *Information and Software Technology*, 50 (9-10), 833-859.

Fayyad, U.M., Djorgovski, S.G., and Weir, N. (1996), 'Automating the Analysis and Cataloging of Sky Surveys', *Advances in Knowledge Discovery and Data Mining*, MIT Press, 471-494.

Feigh, K., and Pritchett, A. (2006), 'Design of Support Systems for Dynamic Decision Making in Airline Operations', *IEEE Systems and Information Engineering Design Symposium*, 136 – 141.

Fu, T.C. (2011), 'A review on time series data mining', *Engineering Applications of Artificial Intelligence*, 24 (1), 164–181.

Garner, J.S., Jarvis, W.R., Emori, T.G., Hogan, T.C., and Hugues J.M. (1988), 'CDC definitions for nosocomial infections', *American Journal of Infection Control*, 1(3), 128-140.

Gould, J.D., and Lewis, C. (1985), 'Designing for usability: Key principles and what designers think', *Communications of the ACM*, 28 (3), 300-311.

Hand, D., Mannila, H., and Smyth, P. (2001), *Principles of Data Mining*, Cambridge : MIT Press.

Hix, D., and Hartson, H.R. (1993), *Developing user interfaces: Ensuring usability through product & process*, USA: Wiley professional computing.

Hong, T.P., Wang, C.Y., and Lin, C.W. (2010), 'Providing timely updated sequential patterns in decision making', *International Journal of Information Technology & Decision Making (IJITDM)*, 9 (6), 873-888.

Jacko, J., and Sears, A. (dir.) (2003), *The Human-Computer Interaction Handbook*, Lawrence Erlbaum.

Jacobson, I., Booch, G., and Rumbaugh, J. (1999), *The Unified Software Development Process*, Addison Wesley Longman.

Kanapeckiene, L., Kaklauskas, A., Zavadskas, E.K., and Seniu (2010), 'Integrated knowledge management model and system for construction projects', *Engineering Applications of Artificial Intelligence*, 23 (7), 1200–1215.

Klashner, R., and Sabet, S. (2007), 'A DSS Design Model for complex problems: Lessons from mission critical infrastructure', *Decision Support Systems*, 43, 990-1013.

Kleindorfer, P.R., Kunreuther, H.G., and Schoemaker, P.J.H. (1993), *Decision Sciences*, Cambridge University Press, 484.

Kolski, C. (1997), *Interfaces Homme-Machine, application aux systèmes industriels complexes*, 2nd edition, Paris : Hermès.

Kolski, C. (1998), 'A call for answers around the proposition of an HCI-enriched model', *ACM SIGSOFT Software Engineering Notes*, 2 (3), 93-96.

Kolski, C., Ezzedine, H., and Abed, M. (2001), 'Développement du logiciel: des cycles classiques aux cycles enrichis sous l'angle des IHM', *In Kolski C. (Ed.), Analyse et Conception de l'IHM. Interaction Homme-machine pour les SI 1*, Paris: Hermès, 23-49.

Larman, C. (2007), *Agile Iterative Development: a Manager's Guide*, Addison-Wesley, Pearson Education.

Lefébure, R., and Venturini, G. (2001), *Data Mining: Gestion de la relation client, Personnalisation des sites Web*, Paris : Eds Eyrolles.

Lemieux, F., and Desmarais, M.C. (2006), 'RUP et conception centrée sur l'utilisateur: une étude de cas', *ERGO-IA*, 11-13.

Lin, H.C., Wu, H.C., Chang, C.H., Li, T.C., Liang, W.M., and Wang, J.Y. (2011), 'Development of a real-time clinical decision support system upon the web mvc-based architecture for prostate cancer treatment', *BMC medical informatics and decision making*, 11-16.

Lepreux, S., Abed, A., and Kolski, C. (2003), 'A human-centred methodology applied to decision support system design and evaluation in a railway network context', *Cognition Technology Work*, 5, 248-271.

Ltifi, H., Kolski, C., Ben Ayed, M., and Alimi, M.A. (2010a), 'Human-centred design approach applied to Medical Dynamic DSS', *The 11th IFAC/IFIP/IFORS/IEA Symposium on Analysis, Design, and Evaluation of Human-Machine Systems*, Valenciennes, France.

Ltifi, H., Ben Ayed, M., Kolski, C., and Alimi M.A. (2010b), 'Démarche centrée utilisateur pour la conception de SIAD basés sur un processus d'ECD, application dans le domaine de la santé', *Journal d'Interaction Personne-Système*, 1 (1), 1-25.

Ltifi, H., Ben Ayed, M., Kolski, C., and Alimi M.A. (2008), 'Prise en compte de l'utilisateur pour la conception d'un SIAD basé sur un processus ECD', *ERGO'IA*, Biarritz, 85-92.

Ltifi, H., Ben Ayed, M., Lepreux, S., and Alimi, M.A. (2009a), 'Survey of Information Visualization Techniques for Exploitation in KDD', *IEEE AICCSA*, Morocco, 218-225.

Ltifi, H., Ben Ayed M., Kolski, C., and Alimi M.A., 'HCI-enriched approach for DSS development: the UP/U approach', *IEEE ISCC'09*, Tunisia, 2009b, 895-900.

Long, J.B., and Denley, I. (1990), 'Evaluation for practice', *tutorial, Ergonomics society annual conference*.

McDermid, J., and Ripkin, K. (1984), *Life cycle support in the ADA environment*, Cambridge University Press.

Mohagheghi, P., Dehlen, V., and Neple, T. (2009), 'Definitions and approaches to model quality in model-based software development – A review of literature', *Information and Software Technology*, 51, 1646–1669.

Muller, M.J. (2007), *Participatory design: The third space in HCI* (revised), In J. Jacko and A. Sears (eds.), Handbook of HCI, 2nd Edition, Mahway NJ USA: Erlbaum.

Murphy, K.P., *Dynamic Bayesian Networks: Representation, Inference and Learning*, PhD Thesis, UC Berkeley, Computer Science Division, 2002.

Peng, Y., Kou, G., Shi, Y., and Chen, Z. (2008), 'A Descriptive Framework for the Field of Data Mining and Knowledge Discovery', *International Journal of Information Technology & Decision Making*, 7 (4), 639 – 682.

Piechowiak, S., and Kolski, C. (2004), 'Towards a generic object oriented decision support system for university timetabling: an interactive approach', *International Journal of Information Technology & Decision Making*, 3 (1), 179-208

Prencipe, A., Davies, A., and Hobday, M. (2005), *The Business of Systems Integration*, USA: Oxford University Press, 392.

Riesbeck, C., and Shank, R. (1989), *Inside Case-Based Reasoning*, Lawrence Erlbaum.

Rumbaugh, J., Jacobson, I., and Booch G. (1999), *The Unified Modelling Language Reference Manual*, Addison-Wesley.

Robert, J-M. (2003), 'Que faut-il savoir sur les utilisateurs pour réaliser des interfaces de qualité?', in G. Boy (Ed.), *Ingénierie cognitive: IHM et cognition*, Paris: Hermès science publications, 249-283.

Royce, W. (1970), *Managing the development of large software systems: Concepts and techniques*, WESCON, technical papers.

Sheng, WH, Chie, WC, Chen, YC, Hung, CC, Wang, JT and Chang SC. (2005),'Impact of nosocomial infections on medical costs, hospital stay, and outcome in hospitalized patients'. *Journal of the Formosan Medical Association,* 104 (5), 318-326

Shi, Y. (2010), 'The research trend of information technology and decision making in 2009', *International Journal of Information Technology & Decision Making*, 9 (1), 1-8.

Singh, Y., and Chauhan, A.S. (2009), 'Neural networks in data mining, Journal of Theoretical and Applied Information Technology', 5 (1), 37- 42.

Somé, S.S. (2006), 'Supporting use case based requirements engineering', *Information Software Technology*, 48 (1), 43-58.

Trabelsi, G., Ben Ayed, M., and Alimi, M.A. (2010), 'Système d'extraction de connaissance basé sur les Réseaux Bayésiens Dynamiques', *EGC'10*, Tunisia.

Zografos, K.G., Androutsopoulos, K.N., and Vasilakis, G. (2002), 'A Real-Time Decision Support System for Roadway Network Incident Response Logistics', *Transportation Research Part C*, 10, 1-18.