Piotr Ziniewicz
Paweł Malinowski
Stanisław Zenon Mnich
Department of Statistics and Medical Informatics
Medical University of Bialystok

CLINICAL DEPARTMENT INFORMATION SYSTEM DEVELOPMENT

Abstract: Advanced informatics systems should reflect the diversity of its users and their needs. An important part of the development process is to study the expectations of its users. This, in result, enables the creation of many different models reflecting specific aspects of such a system.

Introduction

Dynamic development of medical informatics and the growing users requirements, caused informatics systems to evolve into more and more complex applications. Such systems consist of multiple cooperating components located on different machines and communicating with each other using different methods. Programs designed to perform individual tasks become a relic of the past and are slowly replaced by more complex and expanded systems for organized work management. JeNaK management system (from Polish Jednostka Naukowo-Kliniczna – "Clinical and Scientific Unit") is based on the current needs of doctors, researchers, teachers and administrative workers employed at the Medical University of Białystok. Considering the huge problem complexity, the key to the success of the task is to properly build a data structure, logic and application interface. This structure should open the possibility to easily support its expansion in order to meet the changing needs of its users. The whole system should include many possible action scenarios, taking into account the individual needs of each of its user. The aim is to define the structure of the system, meeting the requirements stated above, and its implementation. In this paper, primary collection entity participating in modeled information system will be defi-

ISBN 978–83–7431–266–0 ISSN 0860-150X

ned together with their relational dependencies. The next step will be to define a set of parameters for each entity with their type and data range. Firebird database engine and Embarcadero RAD Studio 2010 was used to implement the system.

Information systems

Last 30 years have brought great changes in healthcare information technology support. The introduction of computers and information systems in healthcare resulted from the need to save on very large and constantly growing healthcare costs. Additional reasons were the increasing amount of information that describes the patient's health state, the need of archiving and granting authorized access to that data, or its transfer between medical units. Now, medical data not only "goes after the patient", but also undergoes the process of wider scientific analysis, for inventing new, more effective and cheaper therapies. Today on the software market there are many hospital information systems that manage and circulate information (van Bemmel, Musel 1997). There are even standards for communication between different systems, such as HL7 (Benson 2010) or DICOM (Pianykh 2008). Many of them contributed for the progress in medicine, patient service at the same time reducing medical costs.

Most of the modern hospital information systems are focused on medical and organizational information circulation, essential for the hospital operation as a therapeutic unit. However, there are no systems that supports potential clinical nature of such "units", including training and research carried out within. The JeNaK system tackles these problems. One of this system's assumptions is large versatility in creating a system handling practically any clinical unit including research projects, without the need for greater modification.

The informatics system is a computer-assisted information system. This information system is defined as a combination of procedures for collecting, processing and transmitting information to support and improve the process of management, decision-making and control. It is a necessary component of any organization or company, including healthcare units.

Construction of the information system

By analyzing the structure of the exemplary information system, one can extract four basic elements (Trąbka/1999, p. 19): hardware, software, data, and users. Hardware is a collection of all specialized devices used for

system construction. Continuous technological progress makes it cheaper and may face increasingly higher user requirements. The essential part is a desktop computer. Usually it is assumed that it has specified parameters, in order to ensure smooth operations, increasing user productivity and comfort. Computers are usually connected together with network devices to exchange information. Today, there are virtually no new computer systems that function on a single computer. Another important part of the information system is software. Software is a combination of programs which supervise work of computers, manage databases and allow the interaction of the system with the users, perform their commands and protect security and confidentiality. Hardware without the end user software is useless. Users are a central element of the information system. The system itself is created to meet their needs. One can distinguish two groups of system users: professionals and end-users. Professionals create and manage quality software and define hardware requirements. End-users are a team of people who work with the finished system. Information system supports data processing to provide ability to enter and process information for potential users.

The process of creating an information system

Creating and managing the rich information system is a fairly difficult task. This is not something new. Already at the turn of the 1960s and 70s, 20th-century programmers first observed fiasco of the major programming projects (Sommerville 2003, p. 71). Software has been delivered late, was unreliable, inefficient and often exceeded the estimated costs of its creation (Brooks 2000, s. 40). Failed projects was a result of incomplete engineering mechanisms and software modeling. Software development quite significantly differs from other manufacturing processes, because the product – software – is "virtual". When producing physical object one can observe the process of its construction and clearly compare with accepted assumptions. Software cannot be touched or seen. Furthermore, programmers are unable to test its functionality for a considerable length of time for the process of its creation is long. Secondly, there are no standards of software creation. In many other areas of engineering, the manufacturing process is checked and tested, and also well known and understood. It is impossible to determine if a concrete process of software creation will lead to the occurrence of specific errors.

Now software is created using objects and classes. These concepts have direct connection with the reality. Biological equivalent class is a species of

animals, and individual objects can be thought of as individuals of these species. Note that some species share with others certain properties (may have a common ancestor). In addition to defined measurable attributes (such as growth or weight), class can have operations (term "behavior" can be used). Classes have property inheritance, so they may use certain operations defined in other classes. Term "entity" instead of "object" is used when turning to database category.

One of the most important achievements in the field of software engineering was the universal modeling language (UML-Alhir 2007). UML Specification 2.2 provides definitions of 14 models that describe many aspects of the system that is created. Each of these models is presented using an appropriate diagram. These models may be divided into two groups in terms of their content and relevance:

1. Structural group of models, these reflect a static system structure using objects, entities, attributes, relationships, operation. These are:
   1.1. class diagram – describes the structure of the system by presenting classes, their attributes and associations,
   1.2. component diagram – describes how the system is divided into components and shows dependencies between them,
   1.3. profiles diagram – describes the so-called metamodel of the system that is a description closer to natural language,
   1.4. structure diagram – describes the internal structure of classes and operations within this structure,
   1.5. implementation diagram – model equipment that is used in the implementation,
   1.6. objects diagram – a complete or partial view of the structure of the system in a specific time,
   1.7. packages diagram – describes the breakdown of the system into logical parts and their relationship to each other,
2. Behavioral group of models – these reflect the dynamic structure of the system, showing the influence of objects on each other, interactions and internal states. In this group one can specify:
   2.1. activities diagrams – show running system and activities of the user step-by-step
   2.2. use cases diagrams – show full or partial functionality of system
   2.3. states diagrams – show the internal state of the object and its changes
   2.4. interaction diagrams – these are rich diagrams that describe the system's behavior. Due to precision, they are created only for critical elements of the system. One of the interaction diagrams is

the communication diagram which shows the interaction between classes as an orderly string message. Sequence diagram shows the interaction between objects as an orderly string message, in addition, it specifies the lifetime of these objects. Diagram of interaction is the general view of the interaction of objects and classes. Each of the nodes in this diagram is a diagram of interaction itself. Time dependency diagram is a specific type of the interaction diagram in which attention is focused on time dependencies.

During the design of JeNaK part of these diagrams were created to make project management more efficient. However, their volume significantly exceeds the volume of this work. Therefore, only the most interesting elements of the system are presented here with their descriptions.

JeNaK functionality

The creation of use case diagrams was the first step during the design of the system. These diagrams show potential users (actors) and system functions (use cases) that can be used by them. Linking actors with functions executed directly by them is marked as a straight lines. Complex functionality of the system can be then split into a series of simpler steps, creating a grid of links between them. There are two types of links between use cases:

1. inclusion marked by «include» – execution of one function always includes an execution linked function,
2. extension marked by «extend» – execution of one function may include an execution linked function.

The example of using these two kinds of calls is shown in Figure 1. Diagram of figure. 1 reflected the patient's visit to a doctor. During the visit it is necessary to conduct an interview with the patient (anamnesis). However, this not always involves a prescription writing. The same combination of operations must be offered by the created system.

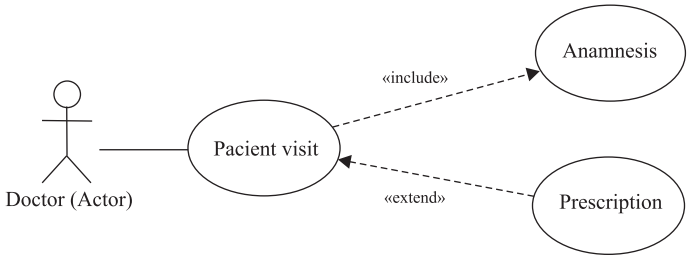

Fig. 1. An exemplary use case diagram

Additionally, scenarios are created for each of the functions that are called by the user. They consist of 5 parts:

1. Participating actors – lists of actors that may call specific functionality,
2. Basic events plot – they describe basic, the simplest course of actions using the sequence listing of interactions between an actor and a system,
3. Alternative events plot – describes all possible aberrations from the basic plot together with their consequences,
4. Time dependency – provides information about how often action is called and how it is time-consuming,
5. Results obtained by actors after finishing use case – gives final effects arising from the execution of a function.

The process of creating a diagram started with conducting surveys concerning duties performed by employees of individual clinical and scientific units. Data was collected from personnel. Below are few examples of many survey questions:

1. What is your role in the unit? (position and brief description),
2. Please list all duties performed in your work,
3. What documents and information you work with and which ones should you archive?,
4. Which steps related to your work you wish to automate? (in the context of the above questions),
5. How does students knowledge verification system look like in the context of your subjects?,
6. How do you collect material for research?,
7. How does staying patient records system look like?

As a result of information analysis gathered from surveys, four types of workers were distinguished based on duties they performed. These types are not disjoint, which means that one employee may perform several function types.

1. Technicians – engaged in administrative and organizational activities and caring for the clinic's equipment. They care about efficient allocation of students into groups and classes, and coordinate this with teachers. An example of a technician is a secretary or a laboratory technician,
2. Scientists – people who conduct research. Often they are interest in unusual cases of patients or rare diseases or complications. Scientists have their acquits in form of scientific publications and gain degrees and awards. Data on the acquits is regularly transmitted to appropriate college structures,

3. Medicals – they deal with patients therapy process, from the patient reception at the ward, forwarded by treatment and commissioning medical check-up, until the end of his/her stay. They deal with both registration and execution, make medical records, register patients observation and perform parts of research. An additional function is to find the unusual cases of illness and inform scientists about them,
4. Teachers – are involved in the didactic process and everything related with it. They are responsible for preparation of student lists and class scheduling, along with reservation of rooms, in which they are assisted by a technician. Teachers also manage "electronic students log" and electronic presence list in accordance with the nature of the imposed teaching obligations. They are also responsible for the preparation of teaching materials to support the learning process and control its effects.

Figure 2 presents a use case diagram of functions related to the administrative job of a secretary employee. Due to the transparency of the diagram it was limited only to the administrative part of the secretarial work. Technicians' duties also spread to the teachers' and scientists' areas. The most common work of a secretary technician is mail management. As it may be seen on the diagram it was split to the three functions: Incoming mail registration, Outgoing mail disposition and Application and requests management. The third one is worth paying more attention. Any application requests funds and results with assets change, therefore it was distinguished as a separate function of mail management. Application for purchase demands to check the funds state and eventually to accept an asset that was purchased. Such function is related to the "Surveillance state funds" and "Keeping the books counting" functions. Despite the fact that all requests and applications that were realized, it automatically changes the state of the funds. There is an additional function named: "Correcting the funds state". This function is a plain "backup" in the case if the real state of funds is different from that counted by system one. All functions of the "Socio-occupational cases" are related to the "Outgoing mail disposition" function due the fact that their files have to be eventually send to the Human Resources Department. "Application for purchase creation" use case scenario is presented below.

1. Actors:
   1.1. Technician,
2. Default event string:
   2.1. System displays "External mail management" window
   2.2. Actor selects "Application and request management"

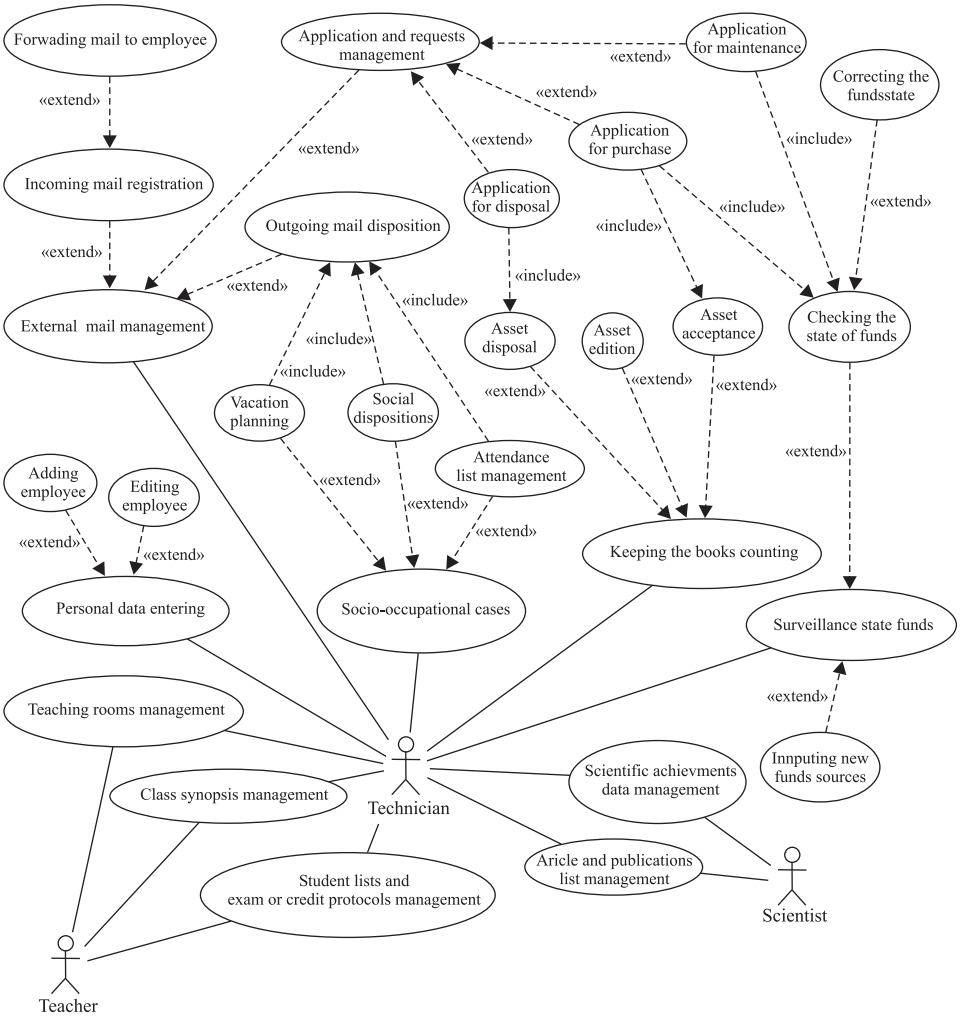

Fig. 2. Secretary use case diagram

2.3. System displays “Application and request management” window
2.4. Actor selects “Application for purchase”
2.5. System displays “Application for purchase” form
2.6. Actor fills subject of purchase description
2.7. Actor acknowledge the value of a subject and fill it to the form
2.8. Actor selects “Add funds source” function
2.9. System displays “Funds state” window
2.10. Actor selects account and closes window
2.11. System displays “Enter amount” dialog window

2.12. Actor enters maximum cash amount to be taken from selected account
2.13. If not all funds sources selected go to step 2.7
2.14. Actor fills reason of purchase
2.15. Actor selects OK option
2.16. System checks if value of subject is less or equal to the funds source amount.
2.17. Actor prints out a paper version of application and send it to the acceptation of head.

3. Alternative event string:
   3.1. Value of subject is greater than funds source amount,
      3.1.1. System displays a warning and returns to the „Application for purchase" form,
4. Time dependencies:
   4.1. Frequency of usage: normal: ~ 5 times a year, pile up season: ~ 5 times
   4.2. Expected pile up: one time a year at the end of year,
   4.3. Typical realization time: $\sim$ 2–5 min,
   4.4. Maximal realization time: unspecified,
5. Values obtained by actors after using:
   5.1. Prepared electronic and paper versions of application,
   5.2. Mail database record,
   5.3. Funds database record (funds reservation).

Overview of JeNaK internals: sample state diagrams

After a range of activities assisted by JeNaK system, it is necessary to define stored data. This is done through the extraction of the set of objects (material or not) participating in the use case diagram. Examples of objects include employee, patient, student (as physical objects) or diet, thesis, score (as intangible objects), etc. All this data is saved in the database as entities. Part of this database, which is related to didactic process was presented in (Ziniewicz, P.; Milewski, R.; Malinowski, P.; Mnich, S. Z. 2010).

Use case diagram focuses on the presentation of the operations supported by the system JeNaK. Functionality of the system is completely separated from the database content. However, experience shows that such separation is quite illusory. Theoretically, every system process stored data according to user's wishes. For the sake of common sense and internal integrity the system attempts to monitor user's actions and warns before,

or eventually denies unauthorized operations execution. Most modern software systems are event-driven, which means that they continuously wait for the occurrence of some external or internal event such as a mouse click, a button press, a time tick, or an arrival of a data packet. After recognizing the event, such systems react by performing an appropriate computation that may include manipulating the hardware or generating "soft" events that trigger other internal software components. Once the event handling is complete, the system goes back to waiting for the next event. The response to an event generally depends on both the type of the event and on the internal state of the system and can include a change of the state leading to state transition. The pattern of events, states, and state transitions among these states can be abstracted and represented as a finite state machine (FSM).

The concept of an FSM is important in event-driven programming because it makes the event handling explicitly dependent on both the event-type and on the state of the system. When used correctly, a state machine can drastically cut down the number of execution paths through the code, simplify the conditions tested at each branching point, and simplify the switching between different modes of execution. Conversely, using event-driven programming without an underlying FSM model can lead programmers to produce error prone, difficult to extend and excessively complex application code.

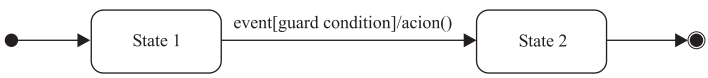

Fig. 3. Sample state diagram

The UML state diagrams are directed graphs in which nodes denote states and connectors denote state transitions. Figure 3 shows sample UML, where one can distinguish:

1. entry point, denoted by solid circle
2. exit point, denoted by circle with solid circle inside
3. internal state, denoted by rounded rectangle labeled by state name
4. state transition from one to another, denoted by arrow, and specially labeled

The arrow label contains information about transition. Label has format event[guard condition]/action(). As mentioned before, event can be generated (triggered) by hardware, software or user. There are also events that occur in specific time (denoted by when(time/date)), or after a specific period of time (denoted by after(duration)). When the event is triggered,

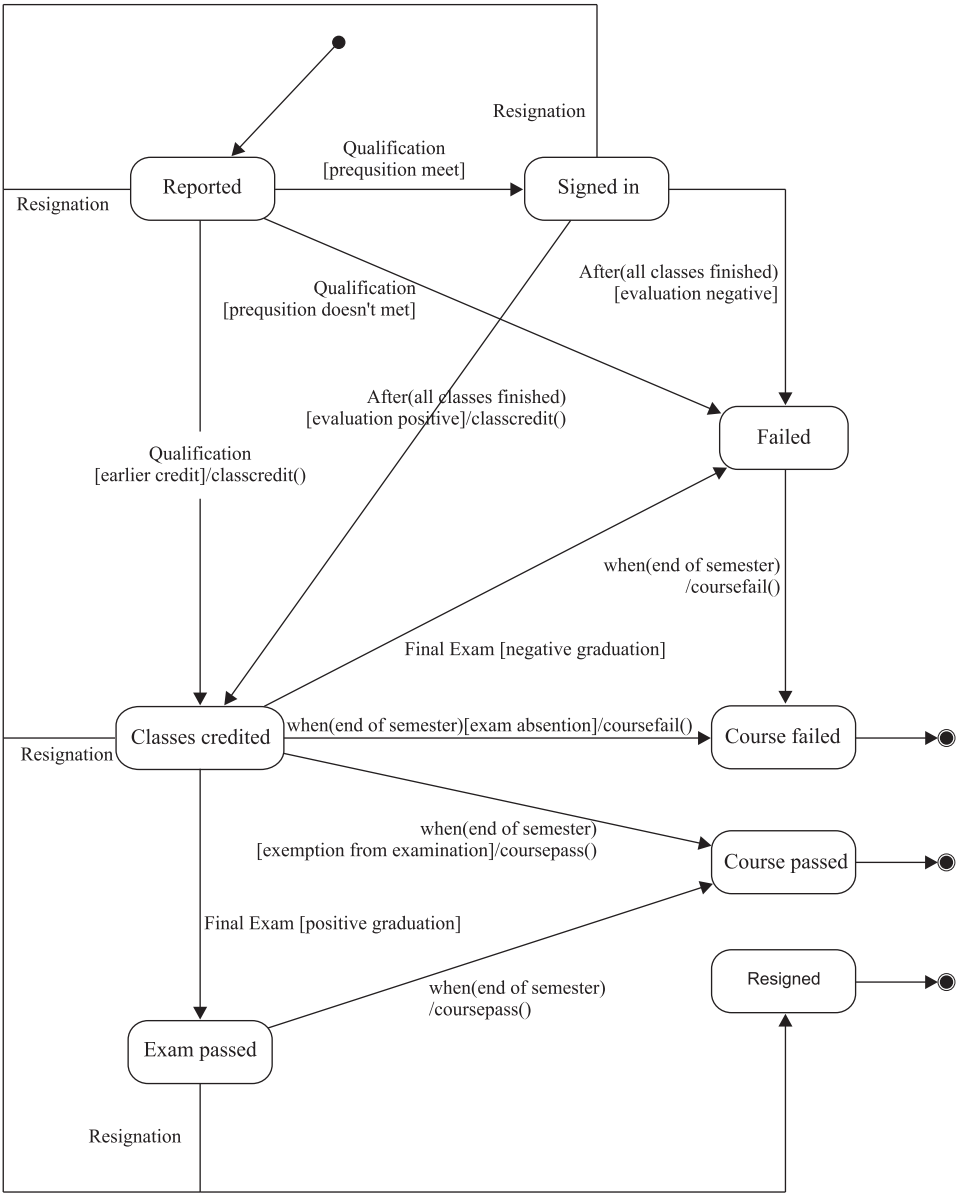

Fig. 4. Student state diagram

guard condition is checked, and if is met, transition to another state occurs, and action() is performed. Latest UML standard (Sommerville 2003) defines many other features in addition, but they will not be used in this work.

State diagram describes the internal state of the selected object in the system, as well as its changes. To create such a diagram it is necessary to

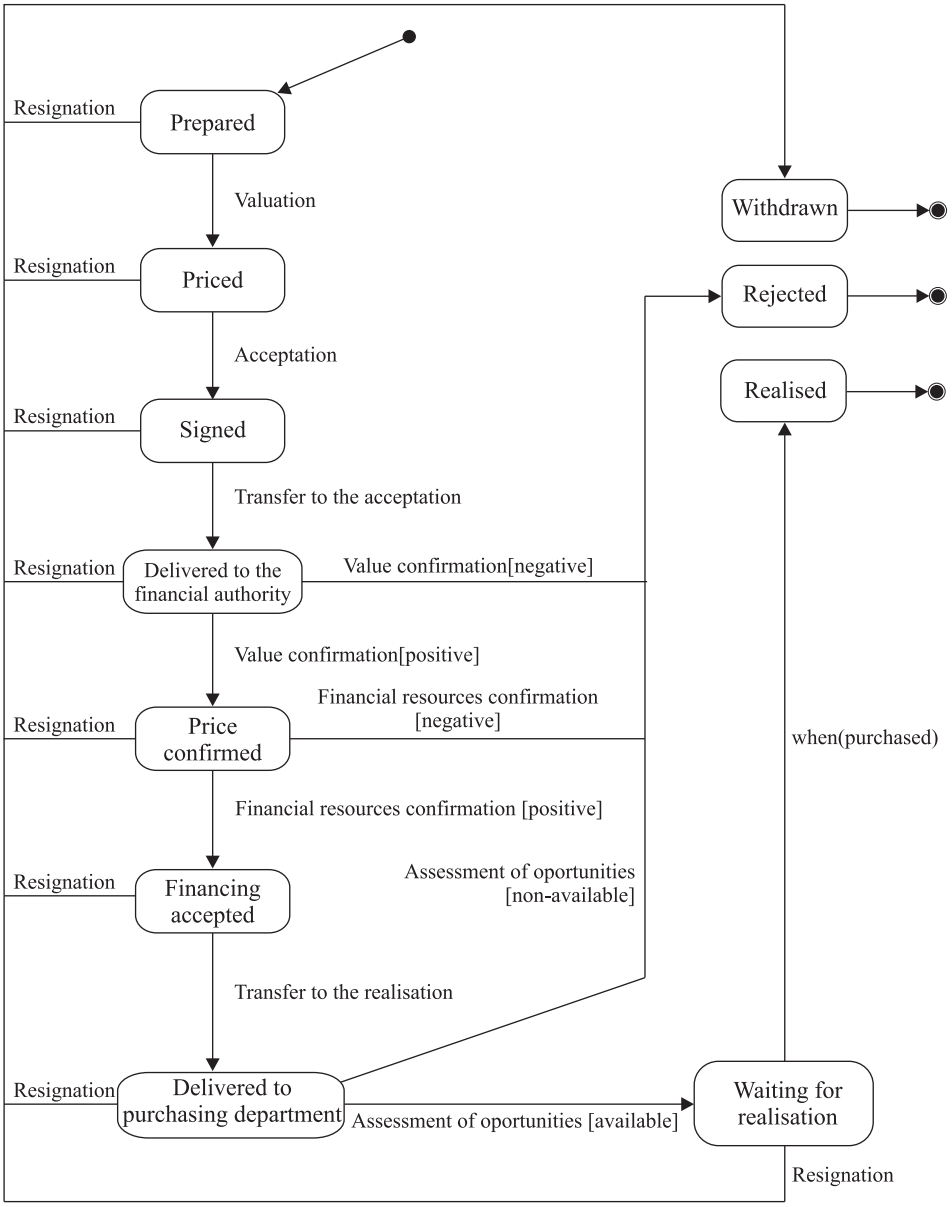

Fig. 5. Application for purchase

track most operations supported by the system, and examine the data on which they operate. Typically, such diagrams are created for data that must be saved for a considerable time, often going through many changes during processing. Usually, for such object, end user will distinguish properties like

status, type, etc. especially where a large number of states and transitions creates fairly dense network of connections.

Inside JeNaK system there are many objects that change their states. Very good example of states diagram can be presented on the base of "student" object, which is presented in Figure 4. To obtain a credit student has to report to the department first. At this stage it is checked if he meets requirements or not. If the student meets requirements, he/she is treated as a legal member of a course and moves to the "Signed in" state. When all classes are completed he/she can change the state to one of the two: "Classes credited" or "Failed". Which of these two will depend on the Evaluation results. At this stage student the waits for an exam and then changes his state to "Course passed" or "Course Failed" accordingly. Worth of attention is the fact that states "Failed" and "Course Failed" are distinguished. The second one is forced by the time and definitive and the first one is more like "continuous". It sometimes happens that the student completed the course before. In such case the during qualification event we have condition: "earlier credit". At the "Classes credited" we have an "exemption from examination" time condition also to omit standard path of the student. At any stage of the path the student can also resign. In such case he/she is treated differently to the one who failed the course because it was his/her conscious decision.

One of the most common documents in secretarial work is an application for purchase. Such document has many stages of preparation which are presented on Figure 5. It has to be prepared, priced, accepted by the head of the department, confirmed by the financial department, etc. At any stage it can be withdrawn by a source user (employee). Subject of the purchase can be priced by the user or by an expert. It does not matter because the application for purchase even when priced has to be confirmed by an expert – once during "Value confirmation" event, and the second time when it is delivered to the purchasing department. It is worth noticing that between "Preparation" and "Waiting for realization" stages the time lapse may be significant. Meanwhile some subjects such as computer equipment may be withdrawn from the market. Prices may also change significantly during this time. "Assessment of opportunities" event has to deal with such cases. At the "Delivered to purchasing department" stage, the expert examines the possibility of purchase and decides if the application will be realized or rejected due to the lack of purchase possibility. Another point of interest are that exit points from "Withdrawn" and "Rejected" which are distinguished. In the first case we deal with a conscious decision of the user, in the second one the user is independent.

Conclusions

Typical hospital information systems have been designed to manage medical and financial aspects of the clinic. They store data related to hospitalized patient, drugs disposal, healthcare insurance organizations accounting, and internal accountancy. Presented work shows a project of a system that manages clinic and scientific information. Unlike typical SSI, focus was put here on scientific, educational and administrative aspects of a single unit.

This system does not replace the whole SSI system, but cooperates with it using standards DICOM and HL7, extending its functionality to include management of research work and finances of an individual unit. It may also cooperate with other systems operating in the administrative units. This extension mechanism and system flexibility enables easy customization of the nature of the research work carried out in a given period, however this requires further research.

REFERENCES

[1] van Bemmel J. H. i Mused M. A. 1997. *Handbook of Medical Informatics*. Berlin, Germany: Springer-Verlag.

[2] Pianykh O. S. 2008. Digital Imaging and Communications in Medicine A Practical Introduction and Survival Guide. Berlin, Germany: Springer-Verlag.

[3] Benson T. 2010. Principles of Health Interoperability HL7 and SNOMED. London: Springer-Verlag.

[4] Sommerville I. 2003. *Inżynieria oprogramowania*. Warszawa: Wydawnictwa Naukowo-Techniczne.

[5] Brooks F. P. 2000. *Mityczny osobomiesiąc*. Warszawa: Wydawnictwa Naukowo-Techniczne.

[6] Alhir S. S. 2007. *Guide To Applying The UML*. New York: Springer-Verlag.

[7] Trąbka W. 1999. *Szpitalne Systemy Informatyczne*. Kraków: Uniwersyteckie Wydawnictwo Medyczne „Vesalius".

[8] Ziniewicz P.; Milewski R.; Malinowski P.; Mnich S. Z. 2010. Informatyczny system zarządzania jednostką naukowo-kliniczną Uniwersytetu Medycznego w: *Współczesne wyzwania strukturalne i menadżerskie w ochronie zdrowia*. Praca zbiorowa pod red. Romana Lewandowskiego i Ryszarda Walkowiaka. Olsztyn: Olsztyńska Wyższa szkoła Informatyki i Zarządzania im. prof. T. Kotarbińskiego.