STUDIES IN LOGIC, GRAMMAR AND RHETORIC 21 (34) 2010

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

THE APPLICATION OF THE JSP METHOD TO THE SYSTEM DESIGNED FOR THE MANAGEMENT OF A CHOSEN MEDICAL UNIVERSITY DEPARTMENT

Abstract: Creation of a system for managing the activity of a Medical University department presents a great challenge for software developers. Such system should reflect many aspects of the activity of such department, including circulation of documents, technical documentation, and patient data. Considerable diversity of data imposes high complexity of the created system. JSP is one of the many methods which is able to cope with this complexity and at the same time can coordinate programmers' team work.

Introduction

Dynamic advances in medical computer science and growing demands of computer users lead to the creation of novel computer systems of increased complexity. Applications designed to realize single tasks are slowly becoming relics of the past, being replaced by more complex systems designated to manage in an organized and guided manner the work of an individual. Such systems contain a number of co-operating components which are located on many machines and communicate with one another using various methods. Then, not only the requirements of project realization, but also proper communication within the entire project should be defined. An additional problem involves changes in the system specifications that may alter its structure. Taking the above into consideration, it can be concluded that the lack of experience in modern methods of software design is one of the most frequent reasons for the failure of the entire process [4].

ISBN 978–83–7431–266–0 ISSN 0860-150X

Software construction process

The basic problem that appears during software construction is its complexity [1]. When the system becomes too complex, the designer ceases to "reign" over all of its aspects. The general understanding of such a complex system in all of its aspects exceeds human possibilities. The division (structuralization) of the system according to the defined criteria seems to be the solution to the problem. However, the problem described by the system has to meet the basic assumption underlying its decomposition, otherwise structuralization of the system will not be possible.

The decomposition means that every element can be described by a sequence of more detailed elements (sub-systems) which also can be split to even simpler elements. This process should be continued until the obtained elements are able to be implemented. As a result of this process, the hierarchical structure is achieved. It is assumed that in such sub-systems, inner-group communication is more dynamic than in the outer one. Expanding the functionality is the natural tendency of computer systems. Decomposition should be conducted in the way that enables easy implementation of a wide range of changes and other modules to the already existing code.

While constructing the computer system it is assumed that the problem to be solved with its use can be decomposed. The software construction is divided into 5 stages:

- Analysis
- Design
- Implementation
- Testing
- Maintenance

The analysis stage includes formalization of the set of system requirements to meet the construction of the functional model of this system, building a model of system interactions with the external world (users, machines) and creation of the information flow control model between the system and its modules. The design stage translates the logical description defined in the previous stage into the description of the physical structure of the system. Based on the physical structure, elements of hardware may be defined as: computers, servers, network media; software: applications, objects, functions and their allocation to equipment components. The implementation stage means coding the algorithms and data using a specific programming language to create a system structure that matches the earlier physical description. In the testing stage software components have to be physically allocated to the hardware units described by the system's struc-

ture. At this stage, proper functioning of the software is tested in the user's environment. Any bugs that appear are fixed. The ultimate user can have access to the system during this phase. Finally, the maintenance stage begins from the moment of the system's delivery to the user and involves: continuous elimination of the bugs that come out during exploitation, expanding the system's functionality according to the user's suggestions and improving the overall performance. Each of these stages should be carried out by an individual team. The stages may overlap or can be implemented consecutively.

The analysis and design of the system (hereinafter referred to design) are the key stages in the process of its creation. Properly conducted, they can significantly reduce the time of software construction, minimize errors, increase reliability, and simplify testing and servicing of the created system. Over the last years, system design has been in the centre of interest among specialists in this field. As a result, many methods that allow for systematic and analytic approach to software design have been worked out. JSP is one of them.

Jackson Structured Programming Method

The JSP method (Jackson Structured Programming, further called the Jackson method) is a well documented and proven method of software design. It is completely independent of the language used. The method was created in 1976 by Michael Jackson and has become a widely used method of design, especially in Europe. In the 70s and 80s of the 20th century it was considered the standard software specification by WHO and the government of the United Kingdom [8]. This method is most frequently used to deal with sequential problems, particularly when the sequence is arranged in time.

The basic principle of the Jackson method is that the design starts with the analysis and modeling of part of the reality in the context of which the system is to work and which is to include. However, no data structures or methods that the system has to perform are specified. The system created with the JSP method contains direct simulation (model) of the reality which has to be specified prior to any designing [5].

Creating a model of reality is generally easier than taking decisions in the design stage. This is because the model frequently exists in reality and is well known to potential users of the system. When modeling a real problem, the software developer should grasp the user's point of view on that problem. Consulting the user may help avoid confusion and numerous

errors in the future. A clear picture of the model defines possible methods and objects of the proposed system.

Another principle of the Jackson method is that an appropriate model of sequential problem is also sequential. The model consists of a sequence of processes that communicate with one another. The sequence should reflect sequentiality of the actual problem. The model should be implemented through specification transformation into an efficient and useful set of processes, adapted to the available hardware. Special attention should be paid to the correct schedule of the processes, and particularly to the fact that a potentially small number of available processor units have to be shared by relatively many separate processes.

The Jackson method-based design process consists of three stages:

- Analysis
- Specification
- Implementation

The first two stages are responsible for the design of appropriate models of the system. The first stage is responsible for creating the data structure. In the second stage, system specification model and program structure model are created. The final stage means encoding of the existing models using a chosen programming language. Sometimes system implementation models are also created; this, however, will not be described here.

Data structure model construction is of major significance in the Jackson method. The design of this model outstrips all other designs, according to the assumption that it is the data structure model that defines the potential space of the system's function models. This is also consistent with the basic design principle to treat the system as a pattern transforming input data into the output data, with very restrictive assumptions as to the form of data structures. These structures can only be represented as a hierarchical tree and are presented graphically as rectangles which can be interconnected by a relationship denoting inclusion. There are three basic types of the relationship: sequence, selection and iteration. They have been presented in Fig. 1.

The sequence (Fig. 1a) means that the structure A consists of components B, C and D. The choice (Fig. 1b) means that the structure A consists of one of the components B, C or D. The iteration (Fig. 1c) means that the structure A potentially contains multiple components C (may not include any of them). It should be emphasized that these basic types of tree-like structures can be mixed with one another to fully reflect the structure complexity. Additionally, each component structure may itself contain multiple smaller parts. This is a consequence of the decomposition rule mentioned

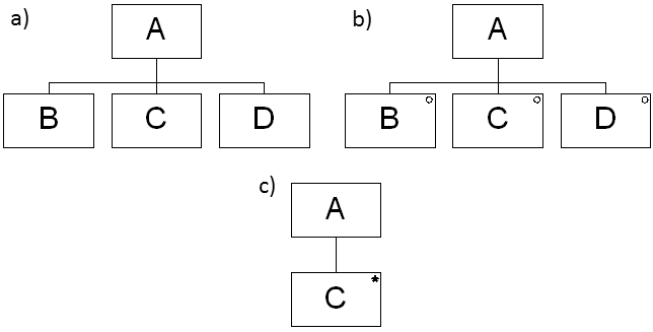

Fig. 1. Tree construction model: (a) sequence, (b) choice, (c) iteration

above. Having developed the full data structure, the application structure model and system specification model should be designed. At this stage, it is assumed that the program structure model inherits from data structure model [6] [7]. This assumption is usually met. If the information consists of separate elements, the procedure processing this information can be divided into appropriate subroutines that process the respective elements. Likewise, processing an alternative means choosing a specific subroutine depending on which component has to be processed. The iterative procedure consists of multiple calls to the same subroutine for each of the components. The appropriate diagrams look exactly the same as in Fig. 1, except that they no longer represent data structures, but the program structure that processes them.

System input data structure

The Jackson method perfectly suits to design a complex system involving a number of interrelated modules. An example of such a system is the application used to support management of educational and research activity of a clinical department. The system operates on a variety of datasets, performing many operations specific to a particular scenario. The scenarios in this case depend on the posts occupied by the system users. Having in mind the enormous complexity of the system logic it is necessary to prepare earlier a detailed documentation in order to avoid complications at later stages of its construction.

To apply the method to design the system that supports management of a clinical department, the following datasets should be specified first [3]:

- all data (entities) inputted to the system (documents that are entered into the system),

- all data (entities) outputted from the system (documents that are presented to the user by the system),
- set of auxiliary data (entity) (used internally as system information store).

Complex information system is split into modules prepared for individual scenarios of system use. The scenarios are related to the posts at which the system is used and so is the list of data ranges used by the information system that supports the clinical department. General data can be divided into 4 categories:

- administrative and financial,
- scientific activity,
- didactic activity,
- medical.

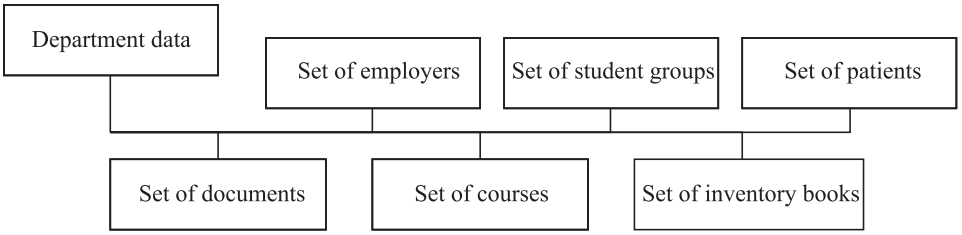

Fig. 2. Overall input data structure of the system

As we can see on the Fig. 2, all of the department data can be divided into six sets. Set of students and courses represents the most of didactic activity of the department and will be described in another article. In this article we will focus on the rest of the sets.

Set of documents consist of many documents data that are used during the department's daily activity. Generally, they may be divided into incoming and outgoing documents. They are represented on the Fig. 3. by "Incoming mail" and "Outgoing mail". It also happened that some documents should be forwarded from one employee to the other (e.g. a draft article can be forwarded to the contributor to implement adjustments). There is an "Internal Mail" on the Fig. 3. for this case. Any mail has a "Header" which gives the date of the mail, its source and destination, document identifier and other common data.

Let us now focus on incoming document data. The most popular mails were split into separated data blocks. They are organized with characteristic for each of them data fields that may be used via dedicated functions (e.g. there may be a function that displays all of the conference invitations with a given subject or destination). There is also a possibility to

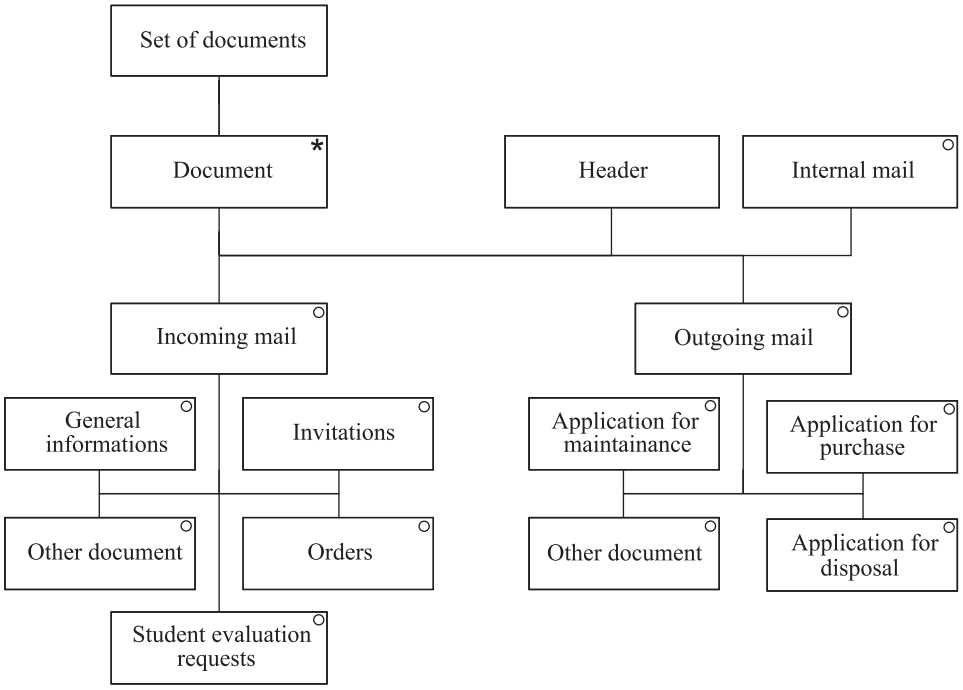

Fig. 3. Input data structure of the documents set

link a "Student evaluation requests" with students from the set of student groups. Then the teacher can run a special function that will display only the requests related with his students. The outgoing mail is frequently used for sending an application for maintenance, purchase or disposal. All of them have to be related to the asset and can modify inventory books. In both cases, besides incoming and outgoing mail, we also have "Other document" data block which includes a common textual memo field which contains the document's content.

In every department there is a necessity to keep the assets record. There is a special data container for this purpose called "Inventory book". On the Fig. 4. one can see an input data structure diagram of a set of such containers. Generally two types of assets can be distinguished: fixed assets and inventories. For every type a separate inventory book is dedicated. Every book has a "Creation time" data field which stores the date when the first record was made in it.

Each inventory book consist of operations set. There can be three operation possibilities: "Acceptance", "Disposal" and "Partial disposal". Depending on the type of asset, different operations are possible and optional data fields can be accessed. Operation "Acceptance" is more universal and

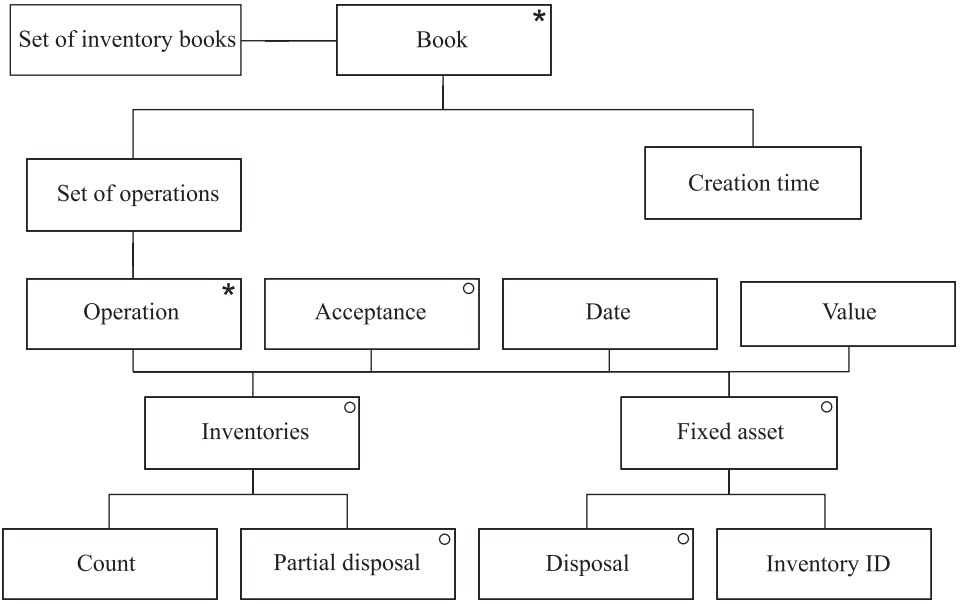

Fig. 4. Input data structure of the inventory books set

appears in both asset type books. Data blocks "Date" and "Value" are general also. First of them denoted the date of the operation and the second one determines the asset value.

In case of the Inventories Book optional operation "Partial disposal" is possible. One inventory can consist of many pieces, therefore operation that dispose only few of them is necessary. Such operation can be also used when one needs to dispose all of the pieces because data field "Count" appears when we operate on the Inventories book. When one substitutes the actual count of the inventory as a count to be disposed, operation of full disposal is made. Data field "Count" is also accessible during "Acceptance" operation made in the Inventories book. In this case it means the count of pieces to be accepted.

During operations on the Fixed assets book operation "Disposal" can be used. There is no need to specify the count because a fixed asset can consist of one element only. In special cases it can be a set of elements of different type (e.g. computer unit). In this case we have a data field named "Inventory ID" which stores unique char string which labels a fixed asset. Operation of "Partial disposal" is not possible here.

One of the most complex data structures of the system is a Set of employers presented on Fig. 5. In this structure not only personal and social data but also all medical, scientific, didactical and administrative data re-

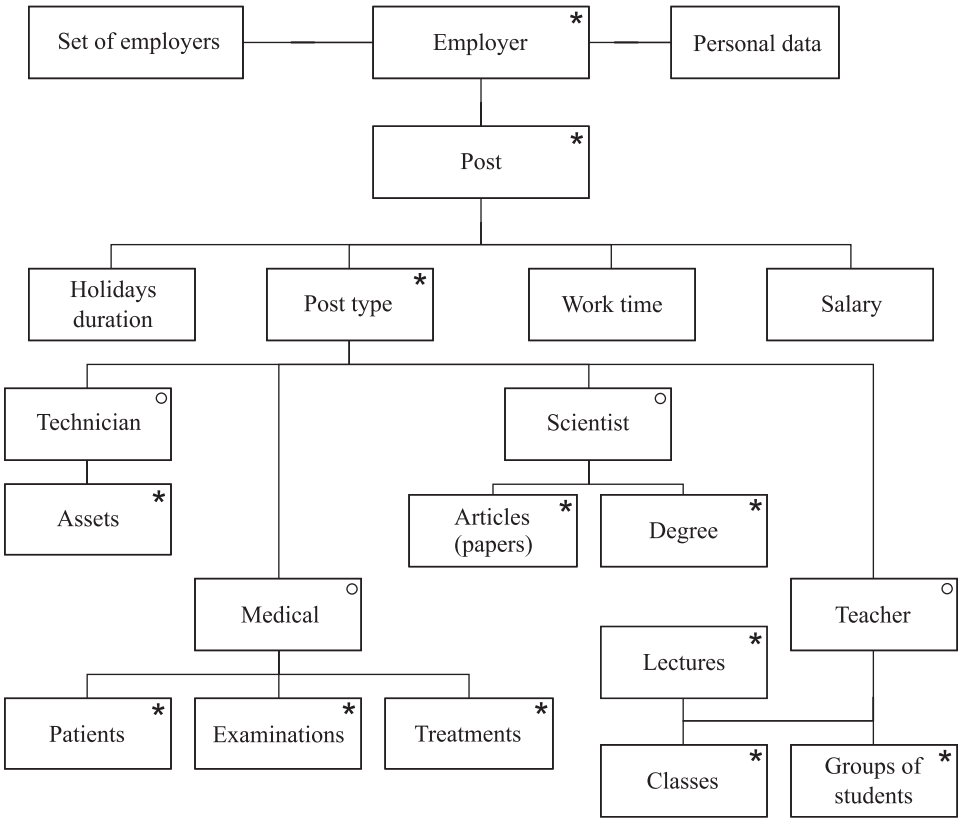

Fig. 5. Input data structure of the employers set

lated to the employee are stored. The most basic data element of this set is the Employer. Every employer has his own "Personal data" block which stores basic data like name, sex, birth date, addresses etc. "Employer" data block determines a person that is employed at the specific post or posts. As one can see on the diagram, every employee can be employed on many posts. The "Post" data block describes a specific post like: secretary, laboratory technician, lecturer, research assistant, cleaner etc. A situation in which one person is employed at many posts is rare but possible.

"Post" data block is bounded to the blocks like "Work time", "Salary" or "Holidays duration", which determines the work conditions at the specified post. Every post is also related to the set of activities bounded to it. It is represented by the "Post type" data block which can be related to one of the four types: "Technician" (administrative activity), "Medical" (medical activity), "Scientist" (scientific activity), "Teacher" (didactic activity). Worth mentioning is the fact that every "Post" can be bounded to several

"Post types". The Post of a secretary should be bounded to technician activity only, but research assistant post must be bounded to medical, scientific and didactic activities simultaneously.

There are several data blocks related to the specific post type, which describes a specified activity. "Medical" data block is related to the set of patients, examinations and treatments that was made or has to be made by an employee. Didactic activity marked as "Teacher" data block on the diagram is bound with "Groups of students", "Classes" and "Lectures" data blocks. Such relation enable the teacher to determine which student belongs to his group and when this group will have classes or lectures. "Groups of students" data block also holds student evaluation and attendance data. Under the "Scientist" data block the information about scientific degrees of an employee can be found. Such information includes obtaining the date of the degree and full article content that was used for. Articles content also is held in such system. "Articles (papers)" data block is designated for this. Such capability enable users to instantaneous access to the scientific production of the whole department. Storing all articles makes the preparation of any scientific survey a simple, and at the same time, a fast task. "Technician" data block is related to the assets that given employee takes care of. It enables the user to check the value of an asset and the date of purchase so that he can determine whether it is still under warranty or no longer.

Patient data stored on the system only includes medical and scientific data, as is presented on Fig. 6. These data do not include management of medicines, bed or insurance billings. Every patient in the "Set of patients" is bounded with a "Personal data" block which contains a simple data like name, sex, birth date, addresses etc. This data determine the person that the patient actually is. Every patient can also have multiple "Contact data" (e.g. to him/her or to the family members). Contact data consist of the name of the person, relationship with the patient, address, phone numbers, e-mail, etc. Most important data block called "Medical history" holds many blocks corresponding to the patient's hospitalizations. Every hospitalization has its "Terms" data block which stores the dates related to it (hospitalization start date, hospitalization end date, pass dates and durations). "Doctor" data block determine which employee takes care of the patient during hospitalization. "Diagnosis" data block stores information about the diagnosis that the patient was admitted to the hospital. During his hospitalization the patient could have been prescribed many different diets. They are all stored in "Diet" data block with information about terms they were in force. Hospitalization bounds to numerous examinations and treat-

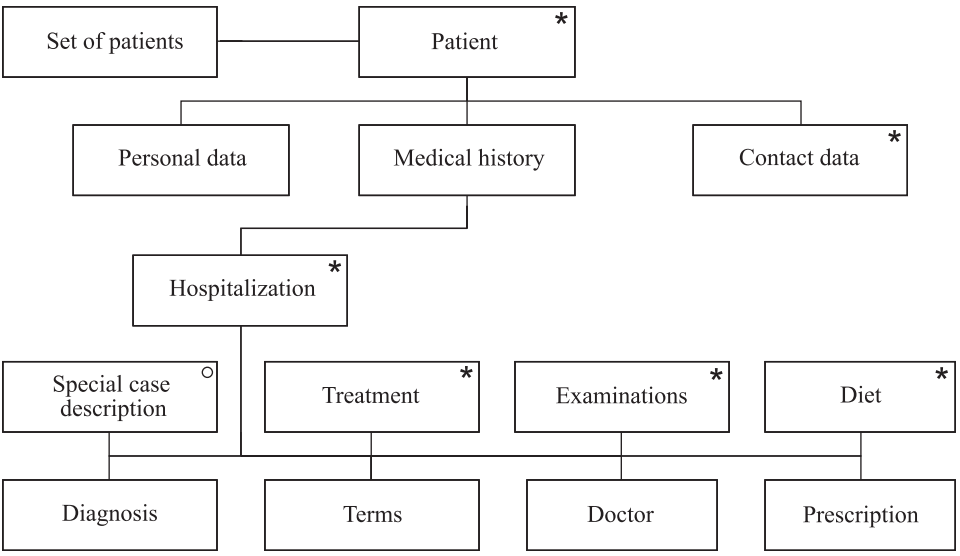

Fig. 6. Input data structure of the patient set

ments. All of them, with their description and results data are stored in the appropriate data block "Examination" or "Treatment". This data draw out all data set of medical history needed for scientific and didactical purposes. There is one more optional data block named "Special case description". It is used to mark a special case that may be interesting for scientist and students. It holds some keywords and case description. Such data are very useful for teachers and scientist and enable them to easily search for the patient's hospitalization that they actually need.

Conclusions

The computer system based on the JSP method ensures compliance with the assumptions made at the stage of analysis and allows tracking its further efficient expansion. The organization of input data structures allows their smooth conversion into elements of the application code responsible for their processing.

The complexity of the system that supports management of a clinical, teaching and research departments makes it impossible to draw up the entire diagram in the current publication. However, the presented fragments prove that with proper effort the presentation of a complete diagram can be feasible.

The elements of the data structure diagram presented above constitute only a small part of the whole project. After correct analysis and decomposition of the problem, the diagram can be completed with new items. Such necessity may occur while discussing the system's functionality with its future users. In such situations, it frequently appears that additional data can increase work efficiency, which should be one of the key benefits of the system implementation [2]. Definition of the diagram for the output and auxiliary data will clarify the tasks of the processes mediating data transformation.

REFERENCES

[1] Adamczewski P., *Zintegrowane systemy informatyczne w praktyce*, Wydawnictwo MIKOM, Warszawa 1998.

[2] Bennett C. J., Computers, personal data and theories of technology: Comparative approaches to privacy protection in 1990s, "Science, Technology & Human Values" 1991, vol 16, no 1, 51–69.

[3] Beynon-Davies P., *Inżynieria systemów informacyjnych Wprowadzenie*, Wydawnictwa Naukowo-Techniczne, 2004.

[4] Flasiński M., Wstęp do analitycznych metod projektowania systemów informatycznych, Wydawnictwa Naukowo-Techniczne, Warszawa 1997.

[5] Jackson M. A., *A System Development Method*; "Tools and Notions for Program Construction", pages 1–26; D Neel ed; Cambridge University Press 1982.

[6] Jackson M. A., *Principles of Program Design*; Academic Press, 1975.

[7] Jackson M. A., *JSP in Perspective*; "Software Pioneers: Contributions to Software Engineering" Manfred Broy, Ernst Denert eds; Springer, 2002.

[8] Ourusoff N., Using Jackson Structured Programming (JSP) and Jackson Workbench to Teach Program Design, "Informing Science" June 2003.

[9] Płodzień J., Stemposz E., *Analiza i projektowanie systemów informatycznych*, Polsko-Japońska Wyższa Szkoła Technik Komputerowych 2003.