

The Approach for User Requirements Specification

Rita Butkiene¹ and Rimantas Butleris²

Department of Information Systems, Kaunas University of Technology,
Studentų g. 50, 3028 Kaunas, Lithuania

¹ ribu@soften.ktu.lt

² rimumbut@if.ktu.lt

Abstract. This paper is a part of research, in which the way the experts of information systems design follow in their work is analysed and specified. The goal of this paper is to present a method for specification of user requirements in the earliest stages of information system development. The technology for specification of user requirements, and the rules, which help to keep correct specification of user requirements, is presented. These technology and verification rules can be implemented in CASE tool to facilitate the specification of information system.

Keywords: requirements specification, information system, enterprise modelling, CASE tool.

1 Introduction

Nowadays there are many methods for the development of computerised information systems (CIS) and project management. The specification of requirements is one of the most important stages in the development of information system. The correct description is one of the principle goals in preparation of specification of requirements. Well-done specification of information system is a significant support to create software with high functionality, low maintenance and adaptation costs.

Many methods rely on graphical specification techniques [1-7]. The reason is that concepts used in the method are easy to understand and use if they are expressed in graphical notation. A formal specification has many qualities. But it is difficult to prepare concrete, formal and understandable to all persons concerned document – requirements' specification. The user must verify all requirements because they are the holders of the requirements. But this is possible if the requirements are presented in the form understandable to user. Formalism used in traditional methods for the requirements' specification is not acceptable to the user if he/she has no special knowledge. These methods are not applicable in the earliest phases of requirements' specification when requirements are informal.

Filled in document forms play an important role in the analysis of user requirements. This analysis helps to reduce the gap between the user and the system analyst. The methods of form analysis [8] are oriented in creation of an entity relationship scheme, but from document forms more information about the needs of user can be acquired. However, not all information used in the CIS development is presented as document forms.

Many powerful tools already exist to assist in system development for instance Oracle Designer/2000 [1,2], Cayenne [9], Rational Rose [7]. But practice shows that these tools sometimes are not useful. The reason is that they use technology of specification of user requirements, which is not adequate to the natural way of analysis of the user needs. In addition, they require high educated and trained specialists to work with them. And the system designer has to work hard for some time.

Information systems designers, experts of information systems development, use their knowledge and experience in their work [10]. They perform the analysis of user requirements and make formal description if user demands it or if it is helpful to them. This way of working is checked in practice and is used successfully.

This paper is a part of research, in which the way the experts of information systems development follow in their work is analysed. The model and process of modelling which deals with earliest stages of specification of user requirements is presented only.

2 Description of model and the way of specification

The task to reduce a gap between user, the holder of requirements, and the designer is not a simple task for the researchers. The one important property of good specification of user requirements is that it expresses these requirements in the notation understandable to the user without additional training. So, what kind of notation satisfies user understanding best?

All people can express requirements in natural language. But expressions in natural language can be full of contradictions, redundancies and ambiguities. Besides, the natural language is not formal, so, finding of shortcomings is complicated enough.

Many different kinds of document forms are widely used for communicating information and knowledge in organisation. A document form reflects a conceptual view of some real-world system and is a partly formal representation of a collection of data. A great deal of conceptual application knowledge is contained in these forms. Therefore, the document forms are familiar to user. So, a solution to use the document forms in the requirements' acquisition, specification, and validation is taken in our approach.

The possibility to extract knowledge about problem domain and to use it for building a conceptual schema has intrigued a number of researchers. The typical approach taken by the researches is to implement prototype system that accepts information requirement statements from user in terms of graphical form designs, or specifications, and examples of filled in forms, and derives conceptual, ER-type schema.

The document forms, which circulate in organisation, can be divided into two subsets: the document forms containing information concerning transactions of business process (1), and the document forms, which generalise an information from the document forms of first subset (2). Examples of document forms from the first subset are orders, invoices, and insurance policies. Examples of document forms from the second subset are financial reports, market reports, and summaries of day sales. In

our approach document forms from the first subset are identified as data resources, and document forms from the second subset are identified as output reports.

In many cases a decision to computerise the information system is taken to get more and in effective way the generalised information about the business. If user can identify what kind of generalised information (output reports) he/she wish to extract from the computerised IS then it is reasonable solution to start the acquisition of user requirements from identification of output reports of computerised system must be developed. Identified output reports must be specified because they are a part of user functional requirements.

In our approach for specification of output report a model of output report is proposed. To make this model understandable to user this model consists of two layers (see Fig. 1): a user-oriented layer and a designer-oriented layer. A user-oriented layer is a graphical design of output report where each graphical element represents a certain concept from output report specification. At first sight it is a copy of document form. Designer-oriented layer is an output report model obtained from the user-oriented layer after several iterations of analysis. This model is a graphical representation of output report's specification where lay out position of each element is as close as possible to the position of the same concept in user-oriented layer. The purpose of this is possibility easier skip from one layer to another. Each element of this model is classified (for example, attribute, group of attributes, value-constraint) and can be related with others elements by relationships (for example, an attribute A is a part of group B). These relationships reflect some semantic dependencies among data, and later are used to build conceptual schema of CIS. The model of output report is used to identify the data resources must be processed by CIS to form specified output reports.

To begin the specification of user requirements from the specification of output reports is reasonable because specified output reports describe the boundaries of the IS must be analysed and computerised. It means, for example, that designers must involve only that business objects in the conceptual model of CIS about which information is presented in the data resources used to form one specified output report at least.

In our approach for specification of data resources a model analogical to the model of output reports is used. This model is similar to the form model from approaches of other researches [8] and is used to derive conceptual schema also.

But not only this schema can be acquired from the data resources. The acquisition of functional requirements, which describe dynamic of CIS, is complicated task for designer. The involvement of data resources in the process of acquisition of these requirements can make this process simpler.

A process of filling in of document forms corresponds to the certain business process. For example, the order is recorded if a customer orders a product; the order is cancelled if a customer cancels it. If information about ordering and cancellation presented in order must be stored in the CIS database, then action "order" and "cancel" are the business actions, which must be supported by CIS because these actions process this information. If ordering information is recorded first and cancellation information can be recorded after only then it means that cancellation of order is possible only if customer performed an ordering action. In this way the precondition to perform cancellation of order was described. Analysing the

alternatives of way how to fill in a certain document form, knowledge about possible sequences of business actions can be acquired.

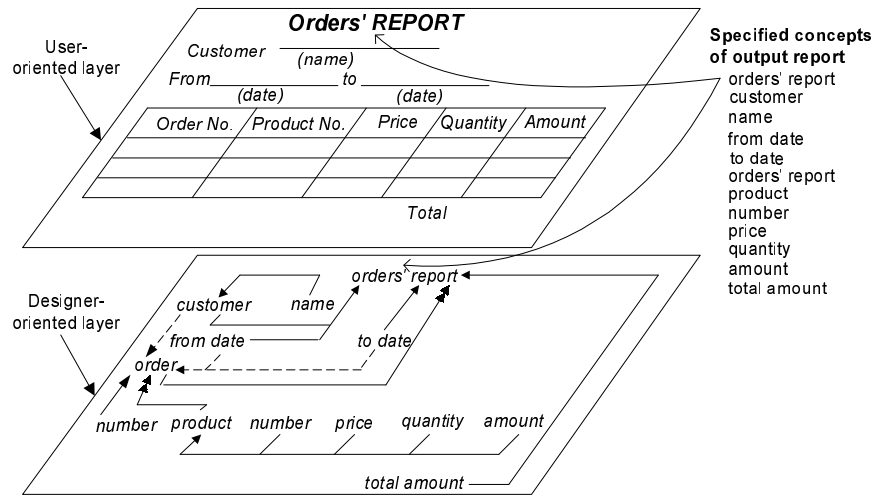


Fig. 1 An example of data resource model

The actors, responsible for performance of specified actions, play an important role in CIS design. Specification of knowledge about them is necessary. The data resources and output reports can be helpful in the acquisition of such knowledge. For example, by asking the user 'who is responsible for filling of certain fields in the document form' knowledge about actors, who are responsible to perform certain action, can be acquired.

In our approach an enterprise model, suggested by R.Gustas [12], was selected to express user functional requirements graphically. Problem is that this model is oriented to designer, not to user. We think that prototype of CIS can be a solution of this problem.

Designers of CIS analyse the document forms to extract user requirements. But the traditional methods (and tools) do not correspond to this way of working. The way the experts of analysis and design of information system follow in earliest stages of specification of user requirements can be described as follow:

1. Specification of functions of CIS.
2. Specification of output reports of CIS.
3. Specification of data resources.
4. Specification of relationships among data resources.
5. Specification of actions processing the data resource.

Function is something a CIS does or needs to do in the future to help to achieve its objectives. *Output report* is a result of functionality of CIS (for example, result of query displayed in screen or printed on paper in specified presentation form). *Data resource* is a resource of data necessary to form the output reports. The data resource can be documents' forms that circulate in organization, verbal reports, and other

information medium. *Relationships* define the way the data resources can be linked. *Action* is an activity, which changes state of data resource.

These five stages presented above correspond to ordinary actions performed by experts in the analysis of the users' requirements. These five stages do not cover a process of specification of requirements fully.

A formal definition of model M of specification of users' requirements to CIS preliminary is defined as tuple $M = \langle Q, F, O, I, T, P, A, S, R, Z \rangle$, where

Q – set of projects,

F – set of functions,

O – set of output reports,

I – set of data resources,

T – set of attributes,

P – set of actions,

A – set of actors,

S – set of states of data resources,

R – set of relationships, $R = R_D \cup R_S$, where

R_D - set of relationships among data resources,

R_S - set of relationships among data resources being in a certain states,

Z – set of IS_A dependencies.

An attribute is a part of output report and/or data resource. *An actor* is something or somebody responsible for carrying out the action and/or interest in outcome of action. A *State* of data resource is described in section 2.5. In the next sections each stage is discussed.

The Oracle CASE Method Entity-Relationship diagram notation [2] (see Fig. 2) is used for graphical representation of model for specification of users' requirements. The model for specification users' requirements concerning computerized functions, output reports and data resources (first four stages of process of specification) is presented in Fig. 3.

First order predicate logic is used to define constrains for each stage of specification.

2.1 Specification of functions of CIS

The specification of user requirements starts from the description of the purpose of CIS. The activity of CIS must be based on the goals, which users of CIS want to achieve in they business with help of CIS. The users of CIS starts description of purpose of CIS identifying which business functions must be computerised. In this way, the part of existing information system must be analysed is identified. Example of function can be registration of orders. In the model presented in Fig. 2 possibility to model relationships among functions, and relationships among actors and functions is not included because of reason do not complicate the diagram. But possibility to model them is taken into account in the earlier research [11].

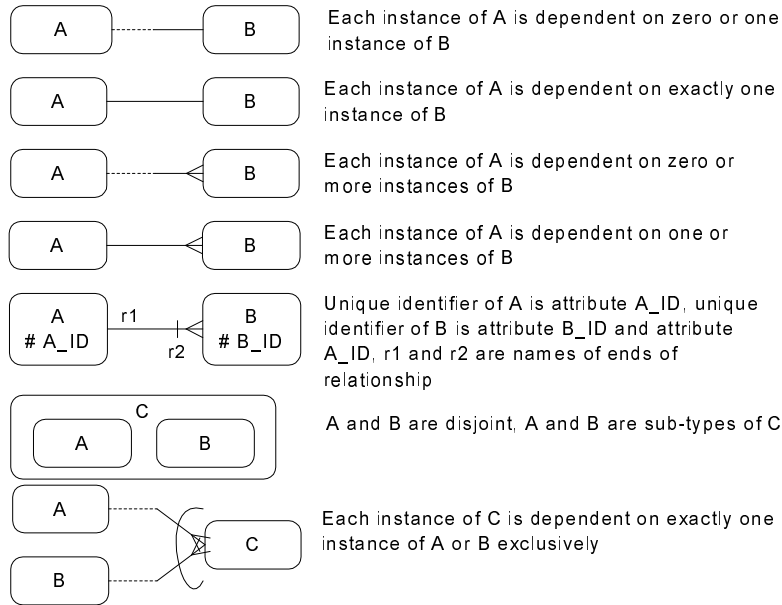


Fig. 2. Notation of Oracle CASE Method Entity-Relationship diagram

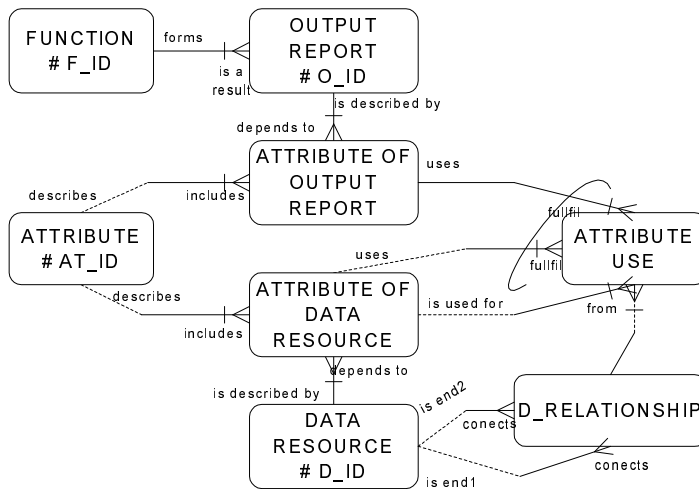


Fig. 3. Model for specification of functions, output reports and data resources

Constraint:

- For each project q at least one function f must be specified which is analysed in project q :

$$\forall q \in Q \exists f \in F [\text{project_function}(q,f)]. \tag{1}$$

2.2 Specification of output reports of CIS

A description of purpose of CIS ends when the output reports are identified to define the results of functionality of CIS.

Constraints:

- For each function f at least one output report o must be specified which is a result of function f :

$$\forall f \in F \exists o \in O [\text{result}(f,o)]. \quad (2)$$

- Each output report o is a result of one and only one function f :

$$\forall o \in O \exists! f \in F [\text{result}(f,o)]. \quad (3)$$

- For each output report o at least one attribute t must be specified which describes this output report o :

$$\forall o \in O \exists t \in T [\text{describe}_o(o,t)]. \quad (4)$$

2.3 Specification of data resources of CIS

Specified output reports define what kind of data resource they need.

Constraints:

- For each output report o at least one data resource i must be specified which is used to form the output report o :

$$\forall o \in O \exists i \in I [\text{use_data_resource}(o,i)]. \quad (5)$$

- For each data resource i at least one attribute t must be specified which describes the data resource i :

$$\forall i \in I \exists t \in T [\text{describe}_i(i,t)]. \quad (6)$$

- Each attribute t describes output report o or data resource i :

$$\forall t \in T [\exists o \in O [\text{describe}_o(o,t)] \vee \exists o \in O [\text{describe}_i(i,t)]]. \quad (7)$$

- Equation (4) can be expressed more precisely. For each output report o and attribute t' , which describes output report o , data resource i and attribute t'' , which describes data resource i , must be specified which is used to form the attribute t' of output report o :

$$\forall o \in O \forall t' \in T [\text{describe}_o(o,t') \Rightarrow \exists i \in I \exists t'' \in T [\text{describe}_i(i,t'') \wedge \text{use_attribute}(o,t',i,t'')]]. \quad (8)$$

- For each data resource i' attribute t' must be specified which describes data resource i' , and is used to form the attribute t'' , which describes output report o , or attribute t''' , which describes data resource i'' :

$$\forall i' \in I \exists t' \in T [\text{describe}_i(i',t') \wedge [\exists o \in O \exists t'' \in T [\text{describe}_o(o,t'') \wedge \text{use_attribute}(o,t'',i',t')] \vee [\exists i'' \in I \exists t''' \in T [\text{describe}_i(i'',t''') \wedge \text{use_attribute}(i',t',i'',t''')]]]. \quad (9)$$

use_attribute(i'', t'', i', t')).

2.4 Specification of relationships among data resources

Specified data resources can be related. If a certain field of data resource is copied to corresponding field of another data resource then it means that relationship between these data resources exists. For example, the invoice to be filled part of data is rewritten from order. Knowledge about relationships among data resources is acquired by analysing the process of filling of data resources. The relationships among data resources must be specified because they evaluate to the relationships among entities (or classes).

Constraints:

- For each data resource i' and i'' , which describes attribute t' and t'' respectively, and attribute t'' of data resource i'' is used to form attribute t' of data resource i' , relationship r between data resources i' and i'' must be specified:

$$\forall i', i'' \in I \forall t', t'' \in T [[\text{describe}_i(i', t') \wedge \text{describe}_i(i'', t'') \wedge \text{use_attribute}(i', t', i'', t'')] \Rightarrow \exists r \in R_D [\text{relate}(r, i', i'')]] \quad (10)$$

- Each relationship r relates two data resources i' and i'' only:

$$\forall r \in R_D \exists! i', i'' \in I [\text{relate}(r, i', i'')]. \quad (11)$$

2.5 Specification of actions processing the data resources

The business actions change the content of information presented in document forms. In our approach the change of this content is expressed as change of state of data resource. A state of data resource can be defined as data resource described by certain set of attributes (not empty) or/and by set of relationships with other data resources. Each state of data resource differs in set of attributes or/and by set of relationships. For example, after ordering action one set of fields was filled in the order, and after cancellation another set of fields was filled in order. The invoiced order must be related with certain invoice. The actions transit data resources to the state in which data resource became suitable to form a certain output report. The actions of user can be of five types:

1. Action of creation – creates the item of data resource in a certain state;
2. Action of transition – transits data resource from state to state;
3. Action of termination – deletes item of data resource being in a certain state;
4. Action of modification – modifies data resource without state changing;
5. Action of search – searches a specified data (in this paper this type of action is not analysed).

Specification of actions processing the data resources defines not only the functionality of information system but gives an opportunity to specify data resources more precisely. For example, relationship between order and invoice reduces ambiguity if it is specified between order being in state “invoiced” and invoice.

Integration of static and dynamic dependencies becomes necessary to get specification of higher quality. Enterprise modelling approach was selected as model having possibility to integrate static and dynamic dependencies. Typical communication action loop is a basic construct of enterprise modelling approach [12]. Communication action loop consists of communication actions and transition dependencies. Graphical notation of communication action and transition dependency is depicted in Fig. 4.

The graphical representation of model for fifth stage is presented in Fig. 5 using Oracle CASE Method Entity-Relationship diagram notation (see Fig. 2).

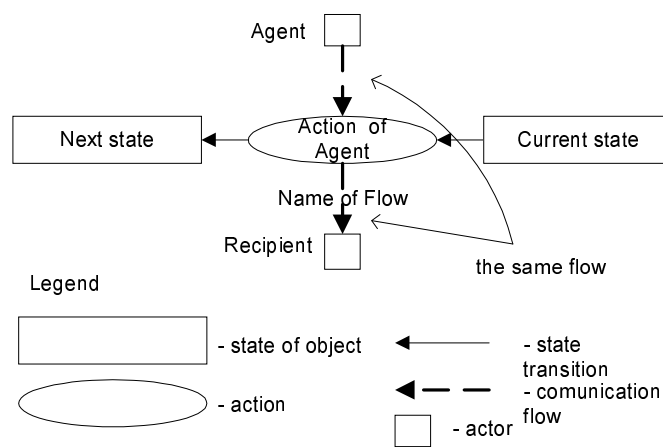


Fig. 4. Communication action and state transition dependencies

Constraints:

- Each data resource i exist in a certain state s . For each data resource i at least one state s must be specified:

$$\forall i \in I \exists s \in S [\text{state_of_data_resource}(i,s)]. \quad (12)$$

- Each state s is specified for one data resource i only:

$$\forall s \in S \exists! i \in I [\text{state_of_data_resource}(i,s)]. \quad (13)$$

- Each attribute t , which describes data resource i and attribute t , describes at least one state s of data resource i :

$$\forall i \in I \forall t \in T [\text{describe_i}(i,t) \Rightarrow \exists s \in S [\text{state_of_data_resource}(i,s) \wedge \text{describe_state}(i,s,t)]]. \quad (14)$$

- Each data resource i being in certain state s must be described by one attribute t at least:

$$\forall i \in I \forall s \in S [\text{state_of_data_resource}(i,s) \Rightarrow \exists t \in T [\text{describe_state}(i,s,t)]]. \quad (15)$$

The states of certain data resource are related by IS_A relationship. For example, an order can be in such states: *Invoiced*, *Not Paid*, and *Paid*. The actions, which

transit an order to these states, are *Invoice*, and *Pay*. These actions are performed consequently. An action *Invoice* transits an order to state *Invoiced*. But really, while an action *Pay* is not executed, the order is transited to the state *Not Paid*, which is a sub-state of state *Invoiced*. An action *Pay* transits an order from state *Not Paid* to the state *Paid*. But after payment an order remains in the state *Invoiced*, also. *Not paid* order and paid order inherit all attributes and relationships, which describe the invoiced order. So, order in state *Not Paid* and order in state *Paid* is an order in state *Invoiced*.

- Each IS_A relationship z relates two states s' and s'' (s' is a specialization of s'') specified for data resource i :

$$\forall z \in Z \Rightarrow \exists! i \in I \exists! s', s'' \in S [\text{state_of_data_resource}(i, s') \wedge \text{state_of_data_resource}(i, s'') \wedge \text{isa}(r, i, s', s'')] \quad (16)$$

- For each relationship r' specified among data resources i' and i'' at least one relationship r'' relating this inputs being in a certain states s' and s'' accordingly must be specified:

$$\forall r' \in R_D \forall i', i'' \in I [\text{relate}(r', i', i'') \Rightarrow \exists r'' \in R_S \exists! s', s'' \in S [\text{state_of_data_resource}(i', s') \wedge \text{state_of_data_resource}(i'', s'') \wedge \text{relate}(r'', i', s', i'', s'')]] \quad (17)$$

- Each relationship r relates only two data resources i' and i'' being in the certain states s' and s'' accordingly:

$$\forall r \in R_S \Rightarrow \exists! i', i'' \in I \exists s', s'' \in S [\text{state_of_data_resource}(i', s') \wedge \text{state_of_data_resource}(i'', s'') \wedge \text{relate}(r, i', s', i'', s'')] \quad (18)$$

- For each state s of data resource i at least one action p must be specified which transit data resource i to the state s (state s is a next state accordingly to the action p):

$$\forall i \in I \forall s \in S [\text{state_of_data_resource}(i, s) \Rightarrow \exists p \in P [\text{next_state}(i, s, p)]] \quad (19)$$

- For each data resources i at least one action p must be specified which creates data resources i in a certain state s :

$$\forall i \in I \exists s \in S \exists p \in P [\text{state_of_data_resource}(i, s) \wedge \text{create}(p, i, s)] \quad (20)$$

- For each action p , which creates data resources i in a certain state s , state s is the expected (next) state of data resource i according to action p :

$$\forall p \in P \forall i \in I \forall s \in S [\text{create}(p, i, s) \Rightarrow \text{next_state}(i, s, p)] \quad (21)$$

- For each action p one and only one agent a' and one and only one recipient a'' must be specified:

$$\forall p \in P \exists! a', a'' \in A [\text{agent}(p, a') \wedge \text{recipient}(p, a'')] \quad (22)$$

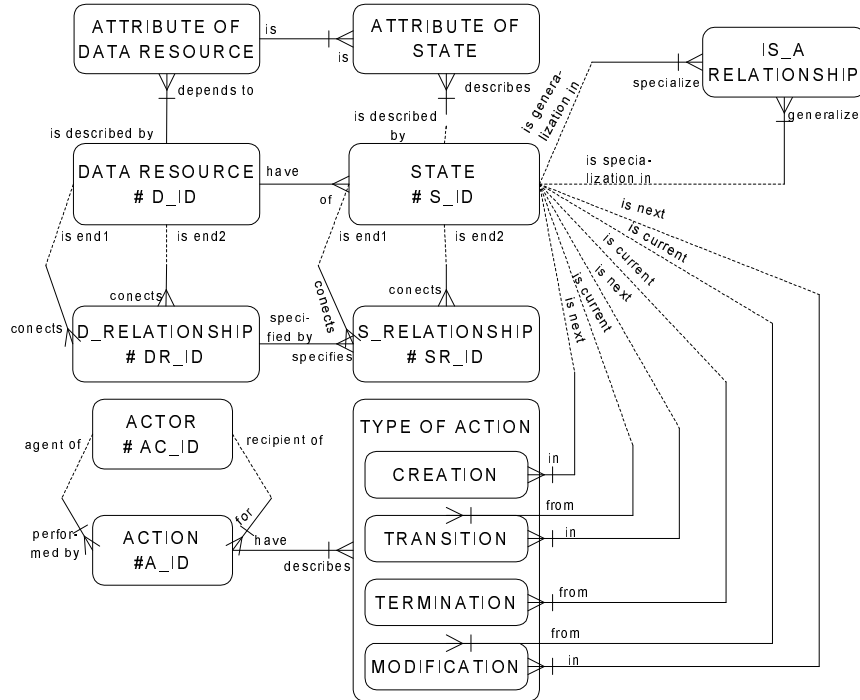


Fig. 5. Model for specification of actions processing data resources

- For each actor a at least one process p must be specified, in which an actor participate:

$$\forall a \in A \exists p \in P [\text{agent}(p,a) \vee \text{recipient}(p,a)]. \quad (23)$$

- For each action p , which transit data resource i from state s' to state s'' , state s' is a current state of data resource i according to action p and state s'' is a next state of resource i according to action p :

$$\forall p \in P \forall i \in I \forall s', s'' \in S [[\text{transit}(p,i,s',s'') \wedge \text{state_of_data_resource}(i,s') \wedge \text{state_of_data_resource}(i,s'')] \Rightarrow [\text{current_state}(i,s',p) \wedge \text{next_state}(i,s'',p)]]]. \quad (24)$$

- For each data resource i at least one action p must be specified which terminates data resource i being in a certain state s :

$$\forall i \in I \exists s \in S \exists p \in P [\text{state_of_data_resource}(i,s) \wedge \text{terminate}(p,i,s)]. \quad (25)$$

- For each action p , which terminates data resources i being in a certain state s , state s is the current state of data resource i according to action p :

$$\forall p \in P \forall i \in I \forall s \in S [\text{terminate}(p,i,s) \Rightarrow \text{current_state}(i,s,p)]. \quad (26)$$

- For each action p , modifies data resource i being in state s , current state of data resource i (and next state of data resource i) before (and after) performance of action p is state s :

$$\forall p \in P \forall i \in I \forall s \in S [[\text{modify}(p,i,s) \wedge \text{state_of_data_resource}(i,s)] \Rightarrow \quad (27)$$

$$[\text{current_state}(i,s,p) \wedge \text{next_state}(i,s,p)]]].$$

2.6 Inference rules

In this section some inference rules are presented. These rules help to identify certain faults in specification of user requirements. Some of them do not give proper conclusion but two or more possible solutions. Right solution can be obtained from the user.

The change of state of certain data resource can change the state of other data resource if these data resources are associated. The consequence of rules (28, 29, 30, 31, and 32) is based on the estimation of states of data resources, the associations among them, and the processing action. The consequence of rules (33 and 34) is based on the estimation of states of data resources, and the processing action. The consequence of rules (35 and 36) is based on the estimation of states of data resources, IS_A dependency among them, and the processing action. The rest rules derive the type of action on the estimation of number of actions specified to certain data resource and on the estimation of role of state (current or next) according to the action.

The rules presented below are proper if states of data resource are disjoint. The associations entitled as bijection dependency and surjection dependency [13] are included in the inference rules because they allow a designer to avoid the holes of semantics. A bijection dependency can be defined as follow: A and B are in bijection dependency if each instance of A is dependent on exactly one instance of B, and each instance of B is dependent on exactly one instance of A. A surjection dependency can be defined as follow: A and B are in surjection dependency if each instance of A is dependent on one or more instances of B, and each instance of B is dependent on exactly one instance of A.

Rules:

- If data resource i' in state s' is in at least one relationship r' of bijection kind or in at least one relationship r'' of surjection kind with data resource i'' in state s'' , and state s' is a current state of data resource i' before performance of action p , and action p according to state s' is not an action of modification then state s'' is a current state of data resource i'' before performance of action p :

$$\forall i', i'' \in I \forall s', s'' \in S \forall r', r'' \in R_S \forall p \in P [[\text{state_of_data_resource}(i',s') \wedge \quad (28)$$

$$\text{state_of_data_resource}(i'',s'') \wedge [[\text{relate}(r',i',s',i'',s'') \wedge \text{bijection}(r')] \vee$$

$$[\text{relate}(r'',i',s',i'',s'') \wedge \text{surjection}(r'')]] \wedge \text{current_state}(i',s',p) \wedge$$

$$\neg \text{modify}(p,i',s')] \Rightarrow \text{current_state}(i'',s'',p)].$$

- If data resource i' in this state is in at least one relationship r' of bijection kind or in at least one relationship r'' of surjection kind with data resource i'' being in

state s'' , and state s' is a next state of data resource i' after performance of action p , and action p according to state s' is not an action of modification then state s'' is a next state of data resource i'' after performance of action p :

$$\begin{aligned} \forall i', i'' \in I \forall s', s'' \in S \forall r', r'' \in R_S \forall p \in P & \quad [(state_of_data_resource(i', s') \wedge \\ & state_of_data_resource(i'', s'') \wedge [[relate(r', i', s', i'', s'') \wedge bijection(r')] \vee \\ & [relate(r'', i', s', i'', s'') \wedge surjection(r'')]] \wedge next_state(i', s', p) \wedge \neg modify(p, i', s')] \\ & \Rightarrow next_state(i'', s'', p)]. \end{aligned} \quad (29)$$

- If data resource i' in state s' is in at least one relationship of surjection kind with data resource i'' being in state s'' , and state s'' is a current state of data resource i'' before performance of action p , then state s' is just a current state of data resource i' before performance of action p , or state s' is a state, which is modified by action p :

$$\begin{aligned} \forall i', i'' \in I \forall s', s'' \in S \forall r \in R_S \forall p \in P & \quad [(state_of_data_resource(i', s') \wedge \\ & state_of_data_resource(i'', s'') \wedge relate(r, i', s', i'', s'') \wedge surjection(r) \wedge \\ & current_state(i'', s'', p)] \Rightarrow [current_state(i', s', p) \vee modify(p, i', s')]. \end{aligned} \quad (30)$$

- If data resource i' in this state s' is in at least one relationship r of surjection kind with data resource i'' being in state s'' , and state s'' is a next state of data resource i'' after performance of action p , then state s' is just a next state of data resource i' after performance of action p , or state s' is a state, in which data resource i' is modified by action p :

$$\begin{aligned} \forall i', i'' \in I \forall s', s'' \in S \forall r \in R_S \forall p \in P & \quad [(state_of_data_resource(i', s') \wedge \\ & state_of_data_resource(i'', s'') \wedge relate(r, i', s', i'', s'') \wedge surjection(r) \wedge \\ & next_state(i'', s'', p)] \Rightarrow [next_state(i', s', p) \vee modify(p, i', s')]. \end{aligned} \quad (31)$$

- If data resource i' in state s' is in at least one relationship r' of bijection kind or in at least one relationship r'' of surjection kind with data resource i'' being in state s'' , and state s'' of data resource i'' can be modified by action p then data resource i' in state s' is modified by action p also:

$$\begin{aligned} \forall i', i'' \in I \forall s', s'' \in S \forall r', r'' \in R_S \forall p \in P & \quad [(state_of_data_resource(i', s') \wedge \\ & state_of_data_resource(i'', s'') \wedge [[relate(r', i', s', i'', s'') \wedge bijection(r')] \vee \\ & [relate(r'', i', s', i'', s'') \wedge surjection(r'')]] \wedge modify(p, i'', s'')] \Rightarrow modify(p, i', s'). \end{aligned} \quad (32)$$

- If state s' is a next state of data resource i' after performance of action p , and state s'' is a current state of data resource i'' before performance of action p then state s^* of data resource i' exist, and this state is a current state of data resource i' before performance of action p , and/or state s^{**} of data resource i'' exist, and this state is a next state of data resource i'' after performance of action p :

$$\begin{aligned} \forall i', i'' \in I \forall s', s'' \in S \forall p \in P & \quad [(state_of_data_resource(i', s') \wedge \\ & next_state(i', s', p) \wedge state_of_data_resource(i'', s'') \wedge current_state(i'', s'', p)] \Rightarrow \\ & \exists s^* \in S [state_of_data_resource(i', s^*) \wedge current_state(i', s^*, p)] \vee \exists s^{**} \in S \\ & [state_of_data_resource(i'', s^{**}) \wedge next_state(i'', s^{**}, p)]. \end{aligned} \quad (33)$$

- If state s' is a next state of data resource i' after performance of action p , and state s'' is a next state of data resource i'' after performance of action p , or if state s' is

a current state of data resource i' before performance of action p , and state s'' is a current state of data resource i'' before performance of action p then at least one relationship r between data resource i' being in state s' and data resource i'' being in state s'' exist:

$$\begin{aligned} \forall i', i'' \in I \forall s', s'' \in S \forall p \in P & \text{ [(state_of_data_resource}(i', s') \wedge \text{state_of_data_resource}(i'', s'') \wedge \\ & \text{[(next_state}(i', s', p) \wedge \text{next_state}(i'', s'', p))] \vee} \\ & \text{[current_state}(i', s', p) \wedge \text{current_state}(i'', s'', p)]}] \Rightarrow \exists r \in R_S \\ & \text{[relate}(r, i', s', i'', s'')]. \end{aligned} \quad (34)$$

- If state s' of data resource i is related with other state s^* of data resource i by IS_A dependency z (s^* is a specialization of s'), and state s' is current state of data resource i according to action p , then state s^* is current state of data resource i according to action p also:

$$\begin{aligned} \forall i \in I \forall s', s^* \in S \forall z \in Z \forall p \in P & \text{ [(state_of_data_resource}(i, s') \wedge \text{state_of_data_resource}(i, s^*) \wedge \\ & \text{isa}(z, i, s^*, s') \wedge \text{current_state}(i, s', p)] \Rightarrow \\ & \text{current_state}(i, s^*, p)]. \end{aligned} \quad (35)$$

- If state s' of data resource i is related with other state s^* of data resource i by IS_A dependency z (s^* is a specialization of s') and state s' is next state of data resource i according to action p , then state s^* is next state of data resource i according to action p also:

$$\begin{aligned} \forall i \in I \forall s', s^* \in S \forall z \in Z \forall p \in P & \text{ [(state_of_data_resource}(i, s') \wedge \text{state_of_data_resource}(i, s^*) \wedge \\ & \text{isa}(z, i, s^*, s') \wedge \text{next_state}(i, s', p)] \Rightarrow \\ & \text{next_state}(i, s^*, p)]. \end{aligned} \quad (36)$$

Two sub-sets of data resources can be defined according to the number of states of data resources to describe the rules presented below:

$I = I' \cup I''$, where

I' – subset of data resources having more than one state,

I'' – subset of data resources having one state only,

$$\forall i \in I' \exists s, s^* \in S \text{ [state_of_data_resource}(i, s) \wedge \text{state_of_data_resource}(i, s^*)], \quad (37)$$

$$\forall i \in I'' \exists ! s \in S \text{ [state_of_data_resource}(i, s)]. \quad (38)$$

- If state s is a next state of data resource i after performance of action p and state s is only state of data resource i then action p creates data resource i in state s :

$$\forall i \in I'' \forall s \in S \forall p \in P \text{ [(state_of_data_resource}(i, s) \wedge \text{next_state}(i, s, p)] \Rightarrow \text{create}(p, i, s)]. \quad (39)$$

- If state s is a current state of data resource i before performance of action p and state s is only state of data resource i then action p deletes data resource i in state s :

$$\forall i \in I'' \forall s \in S \forall p \in P \text{ [(state_of_data_resource}(i, s) \wedge \text{current_state}(i, s, p)] \Rightarrow \text{delete}(p, i, s)]. \quad (40)$$

- If state s'' is a next state of data resource i after performance of action p and state s'' is not the only state of data resource i , then action p creates data resource i in state s'' or transits state of data resource i from state s' to state s'' , or modify data resource i being in state s'' :

$$\forall i \in I' \forall s'' \in S \forall p \in P [[\text{state_of_data_resource}(i, s'') \wedge \text{next_state}(i, s'', p)] \Rightarrow \quad (41)$$

$$[\text{create}(p, i, s'') \wedge \neg \exists s' \in S [\text{state_of_data_resource}(i, s') \wedge \text{transit}(p, i, s', s'')] \wedge$$

$$\neg \text{modify}(p, i, s'')] \vee [\neg \text{create}(p, i, s'') \wedge \exists s' \in S [\text{state_of_data_resource}(i, s') \wedge$$

$$\text{transit}(p, i, s', s'')] \wedge \neg \text{modify}(p, i, s'')] \vee [\neg \text{create}(p, i, s'') \wedge \neg \exists s' \in S$$

$$[\text{state_of_data_resource}(i, s') \wedge \text{transit}(p, i, s', s'')] \wedge \text{modify}(p, i, s'')].$$

- If state s' is a current state of data resource i before performance of action p and state s' is not the only state of data resource i then action p deletes data resource i in state s' , or transits state of data resource i from state s' to state s'' , or modify data resource i being in state s' :

$$\forall i \in I' \forall s' \in S \forall p \in P [[\text{state_of_data_resource}(i, s') \wedge \text{current_state}(i, s', p)] \Rightarrow \quad (42)$$

$$[\text{delete}(p, i, s') \wedge \neg \exists s'' \in S [\text{state_of_data_resource}(i, s'') \wedge \text{transit}(p, i, s', s'')] \wedge$$

$$\neg \text{modify}(p, i, s')] \vee [\neg \text{delete}(p, i, s') \wedge \exists s'' \in S [\text{state_of_data_resource}(i, s'') \wedge$$

$$\text{transit}(p, i, s', s'')] \wedge \neg \text{modify}(p, i, s')] \vee [\neg \text{delete}(p, i, s') \wedge \neg \exists s'' \in S$$

$$[\text{state_of_data_resource}(i, s'') \wedge \text{transit}(p, i, s', s'')] \wedge \text{modify}(p, i, s')].$$

3. Conclusion and future directions

The approach for specification of user requirements in earliest stages of development of computerized information system is presented. The quality of it is that this approach allows reducing the gap between user of CIS and developer. The requirements specified as output reports and data resources are more understandable to user than any graphical or formal specification used in traditional methods. A process of specification proposed for this method is natural and logical, and do not force the developer of CIS to accommodate to methods which need another technology of specification of requirements than he use in his work usually. A specification of requirements obtained in these five stages is not complete but it is good background for evolution. From the specified data resources, states and the relationships the entity-relationship diagram (or class diagram) can be modelled easily than from informal definition. Specified actions are background for the modelling of modules of application. Specified agents of action help to identify in which work places CIS will be used.

The future work is planned on the development of next stages of specification of requirements to CIS. Next stages of the requirements' specification are merged with design stages of information system. One stage of information system design is the data base design. The data base design is based on the specification of data resources, which are modelled up to entities (or object classes) and relationships among them.

Besides, it is important to describe requirements for the user interface. The examples of such requirements are information system menu structure, alignment of screen fields, way of data input, and so on. Requirements of such type are difficult to

express for the user. Only when the user has some experience he can say what he wants. The specification of layout of output reports and data resources can be very helpful in specification of user interface.

This method can be implemented in CASE-tool to facilitate the development of information system.

References

1. Barker, R., Longman, C.: CASE*METHOD: Function and Process Modelling. Addison – Wesley Publ. Co., New York (1992)
2. Barker, R.: CASE*METHOD: Entity Relationship Modelling. Addison – Wesley Publ. Co., New York (1990)
3. Booch, G.: Object Oriented Analysis and Design with Applications, 2nd. ed. Benjamin/Cummings, Redwood City, CA (1994)
4. Coad, P., Yourdan, E.: Object-oriented Analysis. Prentice-Hall, Englewood Cliffs, New Jersey (1991)
5. Martin, J., Odell, J.J.: Object-Oriented Methods: A Foundation. Prentice- Hall, Englewood Cliffs, New Jersey (1995)
6. Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., Lorensen W.: Object-Oriented Modelling and Design. Prentice Hall Int. (1991)
7. Muller, P.A.: Instant UML. Wrox Press Ltd. (1997)
8. Wangler, B.: Contribution to Functional Requirements Modelling. Doctoral Thesis. Stocholm University, Royal Institute of Technology, DSV. Akademitryck AB, Edsbruk, (1993) 189-230
9. ObjectTeam Modelling Guide. Version 5.1.1. Cayenne Software, Inc. (1997)
10. Butkienė, R., Butleris, R.: Computerisation of Conventional Information System Design (in Lithuanian). In The Proceedings of the Conference on Information Technologies. Technologija, Kaunas (1997) 3-6
11. Butkienė, R., Butleris, R.: A Framework for the user requirements specification., Informacinės technologijos ir valdymas, Vol. 2(8). Technologija, Kaunas (1998) 40-53.
12. Gustas, R.: Semantic and pragmatic dependencies of information systems, Monograph. Technologija, Kaunas (1997)
13. Matuzevičius, A.: Topology (in Lithuanian). Lecture notes. Mokslas, Vilnius (1982) 228-236.