ISSN 1392 – 124X INFORMATION TECHNOLOGY AND CONTROL, 2006, Vol.35, No.2

APPROACH FOR IS WORKSPACE DESIGN BASED ON OUTPUT DRIVEN REQUIREMENTS SPECIFICATION

Tomas Danikauskas, Rimantas Butleris

Department of Information Systems, Kaunas University of Technology
Studentų St. 50, LT–51368 Kaunas, Lithuania

Abstract. The problem of information system (IS) workspace design is analysed. Different conceptions for IS workspace structure design are overviewed. Conceptions are based on Oracle CASE method, IDEF and UML application approach. The structure of the output driven requirements specification method is described. An approach to IS workspace design based on information flows specification is proposed. Some suggestions for capturing non-functional requirements for IS workspace are outlined. A case study of information system workspace design is presented.

1. Introduction

The Information System (IS) design process can be separated into three phases:

- IS workspace design;
- IS data base schema design;
- User interface design.

The workspace in this paper is treated as a content of the information system conception consisting of the collection of IS functions and visual layouts of those functions in the menu system.

The second and third phases have a large variety of solutions, both conceptual and implemented, such as IS development tools. The first phase is usually treated as a part of the third phase without paying a lot of attention to IS workspace design. Such an approach works rather well when a small IS, which consists of tens of functions and only few actors, is being designed. However, in such case, finding optimal and fast solutions can become problematic when the IS workspace is being designed for a large scale IS. The objective of this article is to show how this problem could be solved and to explain the importance of the IS workspace definition during the user requirements specification phase.

A specific method is used for user requirements specification. This method, called "Output driven requirements specification method" (***ODRES***), is still in the development phase [1, 2, 3]. The main objectives of this method are to reduce a gap:

- between the user (stakeholder) and the IS designer [1];
- between the analysis phase and the design phase of IS development [4].

The absence of the properties mentioned above could be among the main reasons for possible mistakes or misunderstandings. The first objective was reached by proposing the IS requirements specification process which is adequate to the natural communication between the system analyst and the user [1]. The specialized model was developed for system inputs and outputs analysis with orientation to the stakeholder. This model reduces the gap between users and the IS designer.

The whole process of IS design is easier if analysis and design phases are closely related with each other. The integrity of the development process has influence on the productivity of the process and the quality of design results. Therefore maximizing the fluency of transition from system analysis to design was among the main objectives while working on the proposed workspace design conception.

2. Approaches for workspace design

Two main approaches for IS analysis and design could be discussed [5]:

- Object oriented approach;
- Structured analysis approach.

The best-known representations of each approach are analyzed below from the viewpoint of workspace design.

2.1. Use case

Use case model is one of the main models of UML [6-8]. Several strategies for IS workspace design can be defined depending on the use case model

composition. Depending on objectives, the model construction can be divided into three types [8]:

- User-oriented model – the model focuses on the characterisation of actors and groups of actors.
- Goal-oriented model – the model presents top priority goals that user wants to achieve by using the developed system.
- Work-oriented model – this model specifies user's work situations.

The latter model is most suitable for the workspace, because in common modelling context the model specifies work situations, information objects, actors, properties of attributes and operations suitable for design. Each actor may handle one or more work situations (see Figure 1). There is the 1:1 relationship between the work situation and the workspace in the user interface [9]. In the workspace the actor has access to all information and all tools required in the work situation i.e. communicating with a set of related use cases. The same use case could be reached from different workspaces. Each use case has its own graphic user interface (GUI) elements collected in the workspace.

The decomposition principle could be applied for the use case composition [7, 8]. The use of decomposition gives several advantages. First of all, decomposition of large use cases makes a model more understandable. The reuse principle could be used for decomposed use cases. This possibility is useful in IS development.

As it was mentioned above, the work-oriented model mostly fits for IS workspace design. This model covers all user work situations that must be computerized. In the use case model a separate workspace for each user can be designed. The use cases can be mapped to menu items. The hierarchy of menu items can be added to the workspace, if decomposition is used in the Use case model composition. The use case model isn't adopted for the function hierarchy presentation. If a user has several work situations, the first menu level can be the workspace of the work situation.

The use of the use case for IS workspace design could be defined as a natural use of UML.

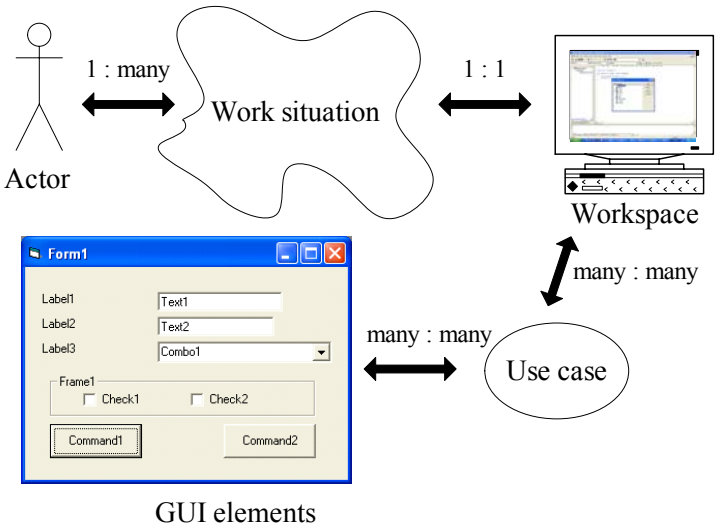


Figure 1. The interaction between use case and user interface design

2.2. Function hierarchy

The process model as a primary instrument for IS workspace design is used in Oracle CASE method [10]. Every unit or actor of the organization has a separate trackway, where the action processed by the activity participant is specified. The workspace is designed by setting separate menu items for each trackway. The main menu item consists of two sub menu items: input forms and output forms.

The function hierarchy can be generated on the basis of process model. The lowest level functions will be realized as input forms or reports. The function hierarchy can also be composed without a process model. R.Barker and C.Longman have presented a full documentation of the function hierarchy construction in their description of Oracle CASE method [10].

Oracle Designer tool also offers the possibility to design the workspace without a process model straight from the function hierarchy. The proposed decision isn't perfect, because workspaces can be designed just by transforming functions of lowest hierarchy into input forms and reports [11].

The function hierarchy is constructed using the top-down or bottom-up strategy [10]. The first function hierarchy model is composed using the top-down strategy. The fullness and overflow of every function hierarchy level is checked by answering to the following questions:

- Have any more functions (except for identified functions) to be performed in order to be able executing the parent function?

- Are there all functions required for executing the parent function?

The hierarchy decomposition is finished when the desirable level of particularity is reached. General guidelines for the function hierarchy composition listed below can help to identify when the hierarchy decomposition has to be terminated. After top-down modeling, the verification of hierarchy can be done using bottom-up modeling.

Business objectives are used in the bottom-up technique. Functions are defined for every objective. Functions detail a higher-level function. The top function is defined as the main objective of business. After the hierarchy composition, the objective list can be revised to check that the function hierarchy presents all business objectives that should be reached. An appropriate adjustment to the function hierarchy must be done, if extra objectives were found.

General guidelines of the function hierarchy composition are presented below:

- According to Oracle CASE method, functions should be decomposed into three to eight subfunctions (if required) [10].
- It is often convenient to arrange functions so that they flow from the life cycle of something.
- Function decomposition must stop when the lowest level functions can be defined as elementary business functions of elementary business transactions, which, when triggered, must be completed successfully. If for some reason a successful completion cannot be achieved, any effects caused by the transaction up to the point of its failure have to be 'undone'.

2.3. IDEF methodology

The IDEF (*Integration DEFinition language)* methods family is used for organization process modelling. One of IDEF conceptions is that all methods must work as a conceptually integrated set of methods [12]. Using the IDEF methods family for IS workspace design IDEF8 is used as a base; the methods IDEF0 and IDEF3 are used as data suppliers for IDEF8.

The IDEF8 Human-System Interaction Design Method isn't directly named as graphic user interface development method. The method is used more to describe the interaction between the user and the system. On the grounds of this specification a detailed GUI design can be done. Any GUI design tool and conception can be used for design.

The IDEF8 process consists of three steps and the whole process is iterative, representing more detailed levels of design [13]. These steps are:

- Define Philosophy of System Operation;
- Design Scenarios of Use;
- Detail Human-System Interaction Design;

Each step increases the level of elaborating the communication between the user and the system.

The first step sets the scope of the system and specifies design objectives that include making strategic decisions and determining success criteria. Critical system functions are identified and specified using IDEF0 functions models, high-level allocations are made and overriding system constraints are identified using IDEF3 process description and the structured language. The product of this mode is an explicit definition of critical system functions and processes characterizing the system's concept of operations. A prioritized list of critical functions and constraints is also produced.

The second step of IDEF8 design centres on role-specific scenarios of use. This mode begins by identifying and classifying various user roles involved in the system. Once the roles have been specified, role-specific scenarios of use can be described using specialized conventions for the IDEF3-based Human-system interface design language (see Figure 2). Next, task/function analysis is performed. Detailed examinations are performed to ensure that the required attributes and services–the required content–are specified. Required modifications revealed through kit reviews and structured demonstrations are also accomplished. The products of the second step are models of various scenarios using IDEF3 models and the results of the task/function analysis.

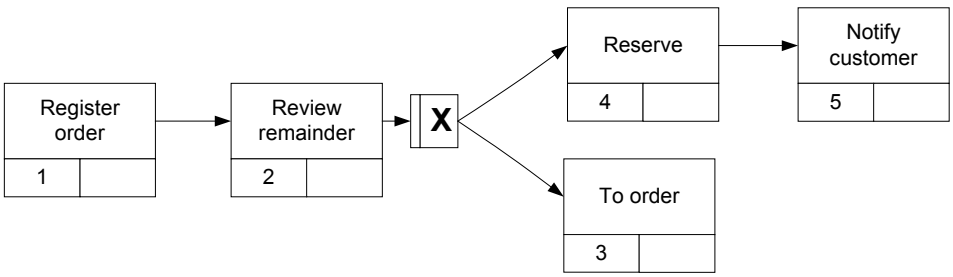


Figure 2. An example of IDEF3 scenarios diagram

Step three begins with more detailed requirements and specifications collected in the dialog guideline definition process. Based on the guidelines, metaphors are selected and models of a detailed human-system interaction are created. When a consensus is reached, human-system interaction mock-ups are constructed to evaluate how well tradeoffs and compromises work in a real world setting. Mock-up development and testing

are used to validate design concepts and to elicit additional requirements. Details of the actual interaction are specified during this step. This may include designing the user's workspace, forms, reports and etc.

The design process of the workspace is not defined in IDEF8, so it is left for user decision. A workspace can be designed for each user role, for each user or can be defined the same for the whole system.

3. Workspace design based on ODRES

Output driven requirements specification method (***ODRES***) is based on the analysis of data flows of the organization [14,15]. In any organization data, flows of incoming and outgoing information exist. The processing of incoming flows creates outgoing data flows. Most of those flows have a defined document or other standardized form, which is commonly used in the organization and can be analysed in the IS analysis and design process [16]. The purpose of IS is to make organization data flows processing and management more effective.

The result of the output driven requirements specification method is data flow specification that can be described as a system consisting of the following models:

Fh – Context model of the information system;

Rds – results / data resources structure model;

Dls – model of links (information flows) between data resources, results and structure of those links;

Dp – results / data resources processing stages model;

Dst – results / data resource state transition model;

El – model of elaboration of links between data resources, results and links between data resources / results states.

The conception of ***ODRES*** CASE tool prototype is presented in Figure 3. The main idea of the conception is the interaction between IS specification and IS design stages. These two stages interact with each other through the specification repository. The interaction presented in Figure 3 is directed only to one side – from the specification to design; it shows the main idea of the method. In reality ***ODRES*** is an iterative process like most of methods for the requirements specification and IS design.

The main objective of this paper, as it was mentioned above, is workspace design. So, for workspace design in ODRES two sources will be used: Fh and Dp models. Next sections explain the composition of ***Fh*** and IS workspace design.

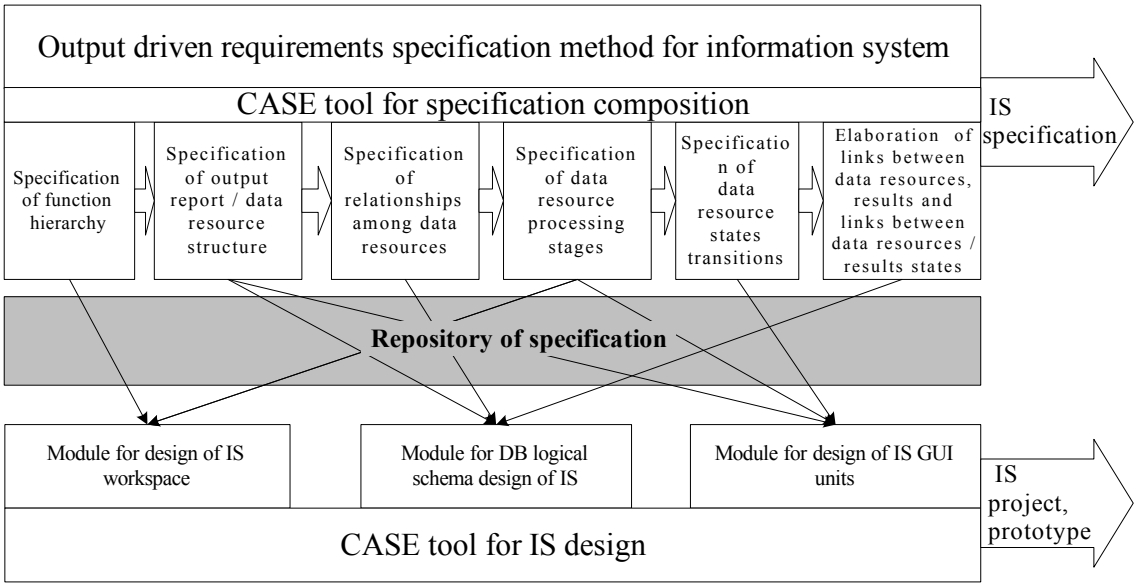


Figure 3. Conception of ODRES CASE tool prototype

3.1. Context description in ODRES

The function hierarchy model was chosen for the context description in ***ODRES*** [17]. The use case model was another considered option. Other models for the context definition [18-21] were rejected from the beginning as less useful or not adoptable for ***ODRES***. ODRES can be subsumed to the class of structural analysis and design methods and this fact had influence on the decision to use the function hierarchy. The structured analysis phase is often seen as intended for the development of so-called output driven requirements specifications. Structured analysis and design is a process-oriented method, because the starting point of the IS development is the analysis of information processes of the system. These processes are described using so-called Data Flow diagrams [5]. As it was mentioned above, the UML is user-centred approach and ***ODRES*** method is business-centred approach. This reason also impacts on the selection of a function hierarchy model for the context description and IS workspace design.

The CASE method proposed by R. Barker and R. Longman was used as a theoretical base for the

functional hierarchy model [11]. In ODRES case, ***Fh*** has been adopted as the information system context model. The notation of the adopted function hierarchy is presented in Table 1.

For every function in ***Fh,*** the name and IS alias must be defined. The name of the function should be a full and descriptive text, using terms familiar to business, but at the same time balance of accurate description and brevity must be found. The IS alias is the short name used in IS for convenience and elimination of using a lengthy name.

Rules and restrictions of the function hierarchy composition will be presented formally and explained by examples. Concepts used in the formal description are presented in Table 2. "Computer component trading" universe of discourse have been chosen for the illustration, ***Fh*** episode of the example is presented in Figure 4. The example presents a computer components trading company that have bulk and retail trading divisions. The bulk division have subdivisions that are responsible for the work with customers and components suppliers. The work with customers is separated by customer location: a local customer and an international customer.

Table 1. The notation of adopted function hierarchy

Graphical notation	Name of component	Description
< Name of top function > **T**	Top function of hierarchy	This function is placed at the top of the hierarchy. The name of the function must express the entire scope of business context under modelling. Only one top function exists in the function hierarchy.
< Name of decomposite function > •••	Decomposed function	A function that consists of one or more other decomposed or elementary functions.
< Name of elementary function >	Elementary function	The function presents one elementary business transaction that must take business from one consistent state to another or not change the state of business at all in case of failure.
	Functions composition relationship (arrow shows parent function)	Hierarchical link between functions. The arrow is directed to the parent function.
O Name of result	Result	An output flow of the IS elementary function.
I Name of data resource	Data resource	Data flow necessary as an input to form results. A data resource can be document forms that circulate in the organization, a verbal message and other information medium that must be computerized.
	Relationship for result / data resource linking with function	Relates result/data resource with an elementary function, which is responsible for result development or data resource processing.

Rules and restrictions of function hierarchy composition are listed below:

1. The function hierarchy model x must include only one top function y and at least two elementary functions y' and y'' at least one result z and one data resource q.

$$\forall x\,[Fh(x) \Rightarrow \exists! y \exists y' \exists y'' \exists z \exists q\,[F(y) \wedge F(y') \wedge F(y'') \wedge O(z) \wedge I(q) \wedge is\_top\_func(y) \wedge is\_elementary\_func(y) \wedge is\_elementary\_func(y) \wedge mhave(x,y) \wedge mhave(x,y') \wedge mhave(x,y'') \wedge mhave(x,z) \wedge mhave(x,q)]] \quad (1)$$

In the sample (see Figure 4) episode we have only one top function "Computer components trading" and more than one elementary function with a data resource, like "New local order" and "Local order" and more than one elementary function with results, like "Month report for local sales" and "Local sales report".

2. For each elementary function x one and only one result y or at least one data resource z must be specified, where y is the result of the function x or the data resource z is processed by the function x.

$$\forall x\,[F(x)) \wedge is\_elementary\_func(y) \Rightarrow \exists! y \exists z\,[O(y) \wedge is\_result(y,x) \vee I(z) \wedge is\_processed(x,y)]] \quad (2)$$

In the sample episode (see Figure 4) all elementary functions have one and only one result or at least one data resource.

Table 2. Description of concepts used in the formal specification

Concept	Description
F – set of functions	The function is some activity the IS does or needs to do in future helping order to achieve its objectives.
O – set of results	The result is an output flow of IS functionality (for example, a query result can be displayed on screen or printed on paper in a specified presentation form).
I – set of data resources	A data resource is a resource of data necessary to form results. A data resource can be document forms circulating in the organization, a verbal message and other information medium.
P – set of actions	An action is an activity that changes the state of a data resource or a result.
A – set of actors	An actor is something or somebody responsible for carrying out the action and/or interested in the outcome of the action.
R_F – set of relationships between functions	Relationships between functions of the function hierarchy define the way the functions can be linked;
R_H – set of relationships between results/data resources	Relationships between results/data resources and functions of the function hierarchy define the way the object can be linked;

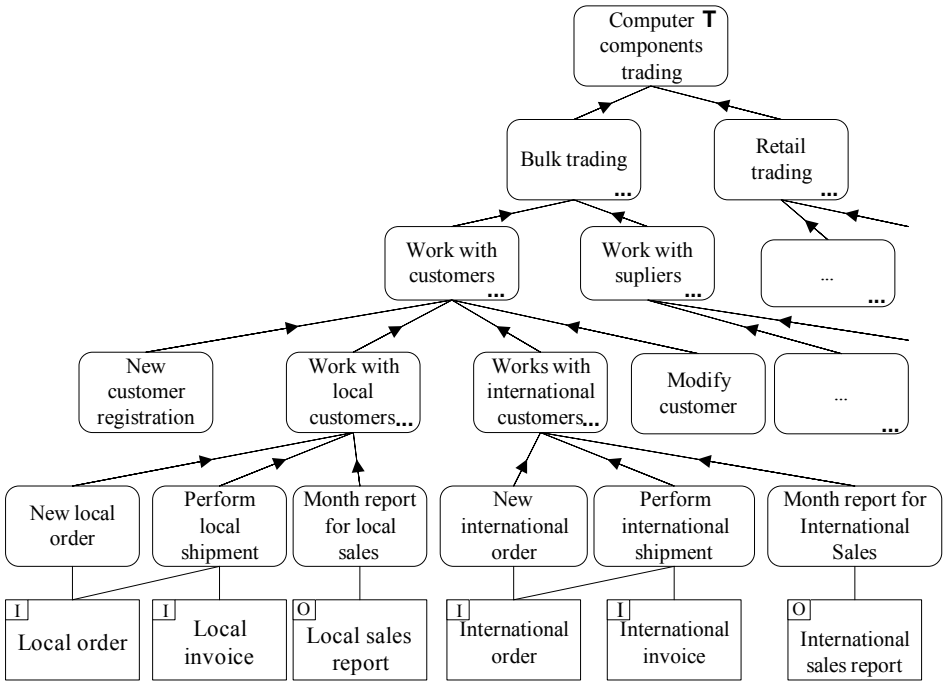


Figure 4. Function hierarchy example

3. Each result *x* must be the result of one and only one elementary function *y*.

$$\forall x\,[O(x) \Rightarrow \exists! y\,[F(y) \wedge is\_elementary\_func(y) \wedge is\_result(y,x)]] \quad (3)$$

The results (see Figure 4) "Local sales report" and "International sales report" are produced only by one function "Month report for local sales" and "Moth report for international sales".

4. Each data resource *x* can be processed by one or more elementary function *y*.

$$\forall x\,[I(x) \Rightarrow \exists y\,[F(y) \wedge is\_elementary\_func(y) \wedge is\_processed(x,y)]] \quad (4)$$

The data resource (see Figure 4) "Local order" is processed by two elementary functions "New local order" and "Perform local shipment". The first function registries a new order, the second function makes the confirmation that the order is complete.

5. Each relationship *x* linking the result/data resource with the function, can link only one result *y* with only one elementary function *q* or only one data resource *z* with only one elementary function *q*.

$$\forall x\,[R_H(x) \Rightarrow \exists! y \exists! z \exists! q\,[F(q) \wedge is\_elementary\_func(q) \wedge ((O(y) \wedge links(x,y,q)) \vee (I(z) \wedge links(x,z,q)))]] \quad (5)$$

In the sample (see Figure 4) all relationships between elementary functions and data resources or results are of binary type.

6. Each function's composition relationship *x* links only two different functions *y'* and *y "*, where *y'* is a child function of elementary or decomposed function

type and *y"* is a parent function of decomposite or top function type.

$$\forall x\,[R_F(x) \Rightarrow \exists!y'\exists!y''\,[F(y') \wedge is\_child\_func(y') \wedge (is\_elementary\_func(y) \vee is\_decomposite\_func(y'')) \wedge F(y'') \wedge is\_parent\_func(y') \wedge (is\_top\_func(y) \vee is\_decomposite\_func(y'')) \wedge links(x,y',y'')]] \quad (6)$$

In the sample (see Figure 4) the child function "Bulk trading" is linked with the parent function "Computer components trading". The other example: the child function "New customer registration" is linked with the parent function "Work with customers". Both examples show that the relationship between functions is only of binary type and parent-child relationships exist between functions. It means that functions of the same hierarchy level can't be related. Example: functions "Bulk trading" and Retail trading" can't be related.

The function hierarchy can be constructed using the above rules and restrictions and general guidelines, which were presented in Section 2.2. The identification of business functions is not discussed in this paper.

The function hierarchy composition in ***ODRES*** is an iterative process. Three phases of composition are defined as follows:

- Context definition phase – in this phase the backbone of function hierarchy diagram (first ODRES model) is constructed. The defined outgoing data flows (results) are linked to hierarchy functions. During this phase the context of IS under development is defined. The analysis and specification process is iterative, so the function hierarchy can be extended with new functions at any moment of applying the ODRES process. Of course, after the extension all specification models must be revised.
- Data resource linking phase – this phase extends the function hierarchy by linking data resources with functions. Data resources were defined during the construction of the model of information flows between data resources, results and the structure of those links. In this phase, a linked data resource can supplement the hierarchy with new functions that were not defined during the first phase.
- Links elaboration phase – in this phase relationships linking results/data resources with functions can be added or modified. This phase will complete after the last model, the model of elaborating links between data resources, results and links between data resources / results states, of ***ODRES*** will be constructed. Results of the elaboration phase can have influence on IS workspace design.

The function hierarchy can grow into a very large-scale diagram, which could be hard to read and difficult to use. That is why the decomposition of the function hierarchy into smaller function hierarchy diagrams is recommended. Criteria for diagram decomposition can be natural business distribution: subdivisions, departments and groups of similar business activities or groups of actors with similar activities.

3.2. Actors and actors groups

In the results / data resources processing stages model (***Dp***) actors and actors groups are defined [1]. This model won't be discussed in the paper. This section just explains how actors can be related with functions. First of all, if the number of actors is large, they could be grouped in order to make the structure of actors less complex. For this purpose, a matrix structure is used. Table 3 presents the actors grouping matrix structure and example. This example relates to the functions hierarchy episode presented in Figure 4.

Table 3. A fragment of the illustrative actors grouping matrix

Actors \ Groups	Bulk sales department	Supply department	Retail trading
Bulk trading director	●	●	
Retail trading director			●
Bulk sales manager	●		
Bulk sales accountant	●		
Bulk sales stockman	●		
Supplier manager		●	
Supplier accountant		●	
Supplier stockman		●	

When the matrix of actors and actors groups is done, we can form the matrix of actors and functions. This matrix is composed using the ***Dp*** model results. If an actor participates in processing of a result/data resource, it means that this actor can perform a function related with this result/data resource. This is the main rule for the actors-functions matrix composition. Table 4 is an illustrative matrix for the example episode presented in Figure 4.

The Matrices of actors grouping and actors-functions could be decomposed into smaller matrices according to the same criteria as the functions hierarchy decomposition.

Table 4. A fragment of the illustrative actors-functions matrix

Functions / Actors /groups	New customer registration	Register new local order	Perform local shipment	Month report for local sales	Modify customer
Bulk trading director	●			●	●
Retail trading director					
Bulk sales manager	●	●	●	●	●
Bulk sales accountant				●	
Bulk sales stockman			●		
Supplier manager					

3.3 Workspace design conception using modified functions hierarchy

IS design can be proceeded after a full set of ***ODRES*** models was collected. The conceptual method of IS workspace design based on ***ODRES*** will be presented in this section. During workspace design the following models and sets of model's components will be used:

- Function hierarchy model and the set of functions.
- The set of actor, which was defined in the Dp model [1].

During the workspace design process the sets of results/data resources, which were specified in Fh, are not used. That is why in the example (see Fig. 6) they are eliminated.

This conception hardly evaluates non-functional requirements. At this moment, security requirements can be evaluated only partially during the workspace design process.

The conception supports two workspace design approaches:

- Business task oriented approach.
- User task oriented approach.

The hierarchy level concept will be used in the workspace design process. The hierarchy level is counted in the top-down direction. The first level of the hierarchy is at the top of ***Fh***, the last level of the hierarchy is in the bottom of ***Fh***. The depth of ***Fh*** is the largest level of the hierarchy found over all branches of the hierarchy.

Business task oriented workspace design. During the workspace design process the following constraints and recommendations have to be considered:

- The amount of different workspaces in IS can be from one to several tens.
- One workspace can be composed of one to 64 applications [22];
- One main menu of the application of a workspace can have up to eight items.
- Pop-up menu of the application can have up to eight items.
- Application's pop-up menu hierarchy should not exceed three levels [22].

A single IS workspace can cover over 200 thousands of elementary business functions, if recommendations for the function hierarchy construction and workspace design are followed. Generally, this conception is suitable for the function hierarchy of eight levels depth. Special rules can be applied for workspace design if the depth of the function hierarchy is larger than eight. A conceptual example of the workspace layout framework, which can be constructed using the proposed conception, is presented in Figure 5. The universe of discourse *Computer component trading* was selected for a case study of IS workspace design rules. Workspace design rules are as follows:

1. If the depth level of ***Fh*** is five or less, the workspace must be designed in the boundaries of a single application where:
   - 1.1 The top function represents an application and the name of the application is alias of the function.
   - 1.2. The second level functions represent the main menu of the application;
   - 1.3. The functions from the third to fifth level represent pop-up menus of the application.

An example presented in Figure 6.a) illustrates the first rule. In case of Fig. 6.a) the workspace was generated for the fragment of ***Fh***, where the function *Bulk-trading department* (see Fig. 4) was defined as a top function of ***Fh***.

2. If ***Fh***'s level of depth is six or seven, the workspace must be designed as a multi-application system. The matrix navigation through applications should be used when the depth level of Fh is seven. Functions of the first and second level represent navigation through the workspace's applications (see Figure 5).
3. If ***Fh*** has a level of depth eight and larger, the system must be designed as a multi-workspace system. The criteria for the decomposition into different workspaces are the same as in the case of the ***Fh*** diagram decomposition and are based on natural business distribution: subdivisions, departments, groups of similar business activities or groups of actors with similar activities. If the

decomposition of ***Fh*** into a few separate ***Fh*** was done before, it could be used as a basis for the IS decomposition into workspaces.

The functions *Bulk trading*, *Retail trading* (see Figure 4) could be the top functions for new workspaces in case of workspace decomposition.

4. In some cases the functions of a higher level of the hierarchy can be ignored during the workspace design process. Every higher-level function generalizes its sub-functions. Sub-functions of decomposite function(s) can be linked with a higher-level function during the workspace design process, if the elimination of function(s) doesn't add ambiguity to the workspace. This rule can be applied not for all level functions of ***Fh***, but just for single functions of ***Fh***. Alias of sub-functions of eliminated functions can be corrected according to the alias of parent functions, if the rule four was applied in the design process. The rule four can help optimising the workspace layout.

The example in Figure 6 illustrates the application of the rules. In case a) the rule four wasn't used during a particular workspace design, in case b) the rule four was applied to functions: *work with local customers* and *work with international customers*.

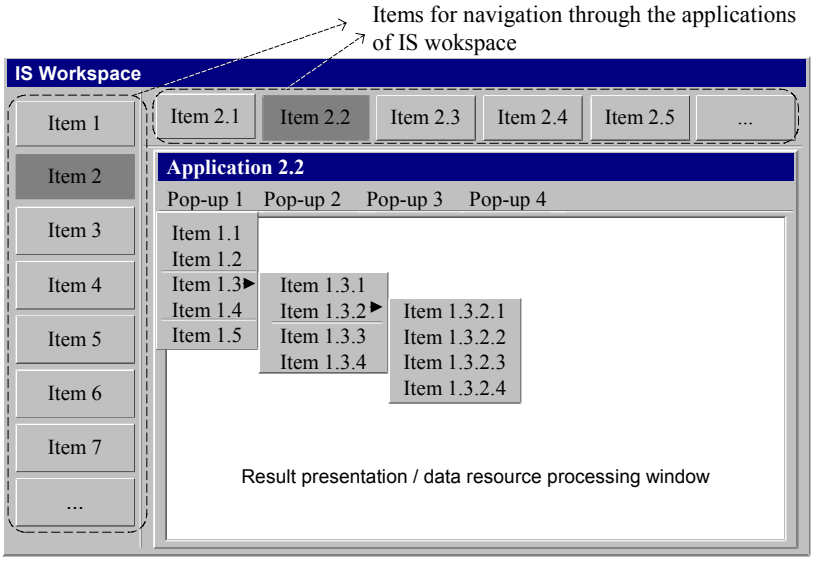
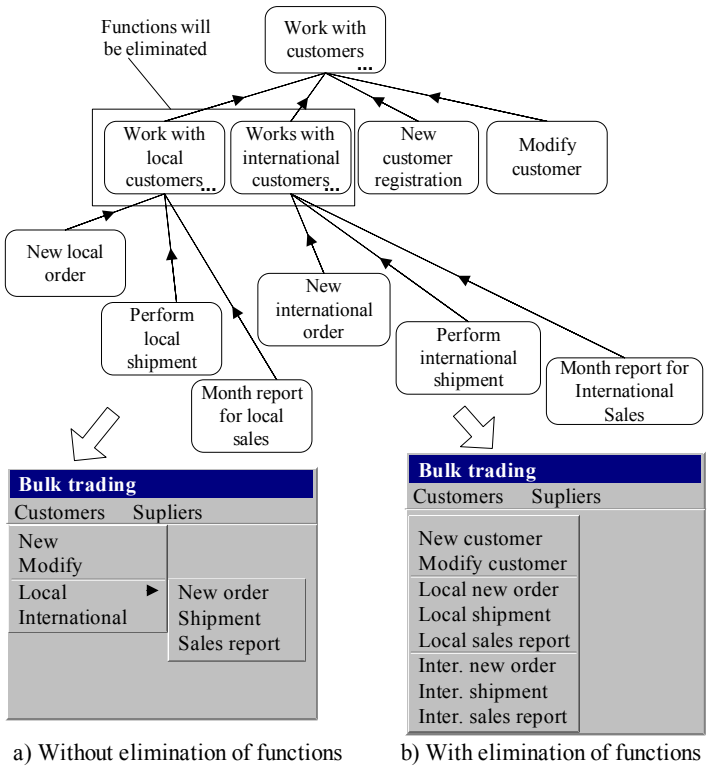


Figure 5. The IS workspace layout conception



a) Without elimination of functions b) With elimination of functions

Figure 6. The cases of workspace design

The elimination of functions in the design process reduced the depth of pop-up menu, which, as it was mentioned before, helps to optimise the workspace layout. The reduction wasn't so essential in this example, but it perfectly illustrates the purpose of the rule.

The universal workspace for any IS can be designed using rules and recommendations presented above.

User Task Oriented Workspace Design. The user tasks oriented workspace can be designed using information, which is accumulated in actors grouping and actors-functions matrices. The actor *Bulk-trading manager* (alias: director) was defined according to Figure 4. The defined actor will be used to illustrate steps of the user task oriented workspace design.

Steps of the user task oriented workspace design process are as follows:

1. Create a list of functions candidates used by each user specified in ***ODRES***. Functions for each actor (user) can be found in the actors-functions matrix.

The list of initial functions of *Director* can be formulated (just a fragment of ***Fh***, which is presented in Figure 6 was analysed). Functions of *Director*: *Month report for local sales, Month report for international sales, New customer registration, Modify customer information*.

2. Find the top function and depth of the user's function hierarchy. The top function of the user is ***Fh*** function, from which all elementary functions of the user can be reached. The top function of the user can be different from the top function of ***Fh*** of ***ODRES***. *Function hierarchy of user* is a subset of ***F***, that includes the user's top function, user's elementary functions and all decomposed functions linking top and elementary functions of the user. The top function and the depth of the user function hierarchy can be found searching ***Fh*** in the bottom-up direction starting from the elementary functions of the user.

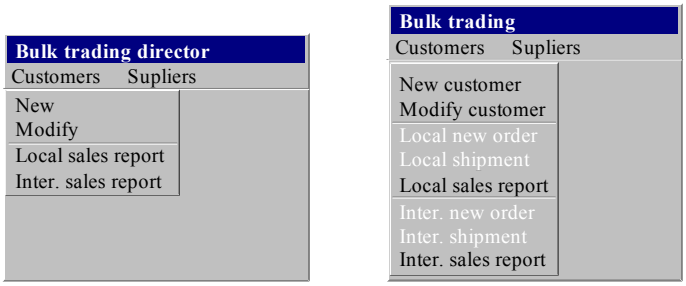
The top function of the user *Director* is *Bulk trading*.

The depth of user's ***Fh*** is 4.

3. Design a user task oriented workspace. The user workspace could be designed following rules and recommendations of business task oriented workspace design and user function hierarchy.

An additional rule for user task oriented workspace design: Items of a workspace can be eliminated if an item has only one sub-item. After the item elimination, a sub-item is attached to the item of a higher hierarchy level.

Figure 7a illustrates *Bulk sales director* workspace. User's *Director* menu consists of four elementary functions defined during the second step. The items elimination rule was applied to eliminate menu items that had have only one sub-item.



a) User task oriented workspace

b) Security requirements applied for Business task oriented workspace

Figure 7. User task oriented workspace design

The workspace design could be performed for a user group. Users should be organized into groups according to the defined criteria. Like it was mentioned above, criteria can be similar to the ones used during the ***Fh*** decomposition. The process used for designing the users' workspace also could be used for designing the user groups' workspace.

A non-functional security requirement – "in a workspace the user can perform only her own tasks" could be applied to the business task oriented workspace. User login functionality should be added to the workspace. When the user logs into the system, only applied items of the workspace, which are included in the list of user's elementary functions, will be enabled for this user. Figure 7b illustrates user's *Director* workspace when non-functional security requirements were applied to the business task oriented workspace.

4. CASE tool prototype of workspace design

The conception of workspace design presented in this paper is implemented as a specialised module of CASE tool prototype. It will include the functionality for the ***ODRES*** specification composition and IS design based on specification results. At this moment, an initial version of business task oriented workspace design is developed. The developed solution is based on MS Visio 2002 template (see Figure 8) with the stencil developed for ***Fh*** composition, and also on the extended functionality that could be reached via menu items. The composed ***Fh*** can be saved not only as a

drawing, but also as a set of linked components stored in the repository. The MS Access database for metadata storage is used. The use of the repository-developed module can offer the functionality for the workspace framework generation.

The workspace generation results using a MS Access report are presented (see Figure 9). The workspace is presented as a function tree that is accessible for users of a given workspace. The documentation for all workspaces can be generated in such a way. The user tasks oriented workspace generation is under development now. Finally, all ***ODRES*** models and design phases will be developed as a set of logically linked Visio templates [23, 24].

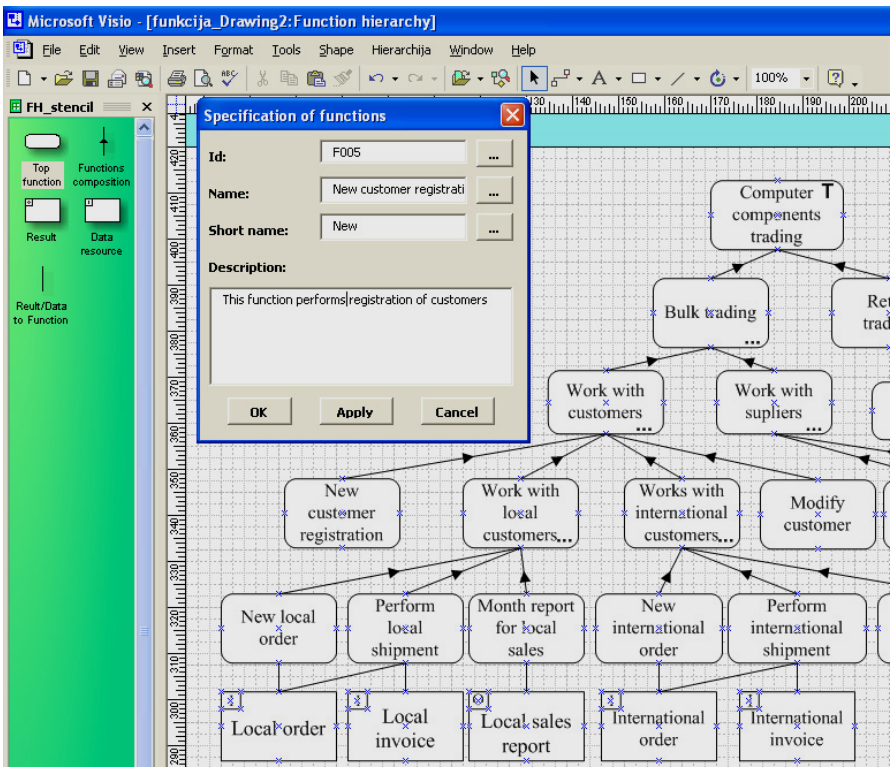


Figure 8. CASE module for the function hierarchy composition

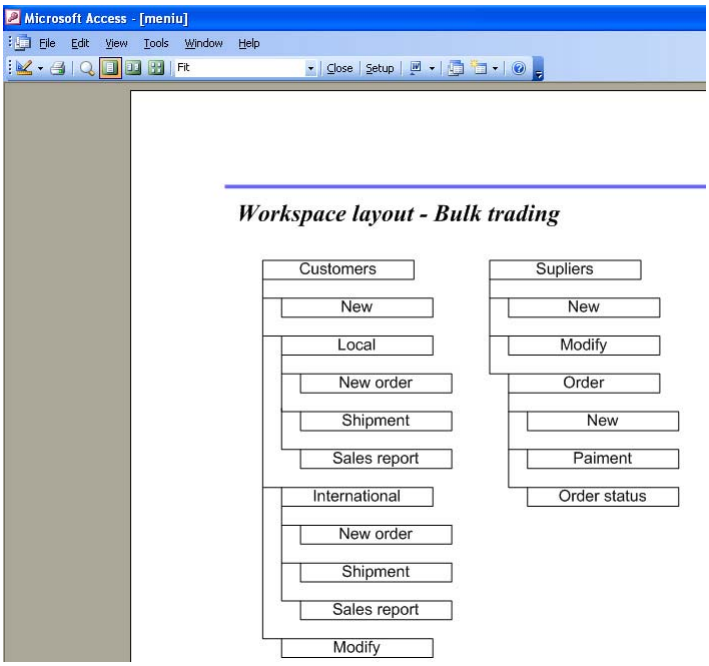


Figure 9. The results of workspace generation

5. Conclusions

Several techniques for workspace design were reviewed in this paper. These techniques represent object oriented and structured analysis approaches for IS design.

The description of the output driven requirements specification method and requirements specification process is presented. This method is based on a structured analysis, therefore the function hierarchy was chosen for the IS context description and workspace design. The function hierarchy model was enriched by Results/data resources. It enables to relate functions with actors and groups of actors. Rules and restrictions were formally described for the functions hierarchy model composition.

The IS workspace design conception based on the Function hierarchy model and Results/data resources processing stages model is composed. The proposed conceptual IS workspace design process was presented in detail. A case study of workspace design using the proposed framework was carried out. It has shown that the conception works in a real problem domain.

The proposed conception is advisory and can be applied according to the given universe of the discourse or specific characteristics of IS under development.

The conception includes a universal framework of the IS workspace layout, but quite a few existing GUI elements used in traditional workspace design are not evaluated [25, 26]. This will be accomplished in future works.

The workspace design process is implemented as a module that includes autonomous functionality of the CASE tool. It allows a partial automation of the design process and results of a better quality can be reached faster. The tool works as a workspace design process wizard proposing solutions or suggestions for a designer.

References

[1] **R. Butkienė, R. Butleris.** The Approach for the User Requirements Specification. 5th *East-European conference ADBIS*'2001, *Research Communications, Ed. by A. Čaplinskas, J. Eder, Vilnius,* 2001, 225-240.

[2] **R. Butkienė, R. Butleris.** Verification Rules of Computerised Information System Model with Respect to Data Resource Processing. *Informatica. Vol.*12, *No.*3, *Mokslo aidai, Vilnius,* 2001, 347-372.

[3] **R. Butkienė, R. Butleris**, T.Danikauskas. The approach to consistency checking of functional requirements specification. *The* 6th *World Multiconference on Systematics, Cybernetics and informatics. Proceedings of International Conference, Vol.*18, *Orlando, USA*, 2002, 67-72.

[4] **H. Saiedian, R. Dale.** Requirements engineering: making the connection between the software developer and customer. *Information and Software Technology, Vol.*42, *Elsevier*, 2000, 419-428.

[5] **A. Solvberg, D.C. Kung.** Information systems Engineering. *Springer-Verlag, Berlin Heidelberg New York*, 1993.

[6] **L.A. Maciaszek.** Requirements analysis and system design developing information systems with UML. *Addison-Wesley*, 2001.

[7] **P. Muller.** Instant UML. *Wrox press Ltd*., 1997.

[8] **M. Van Harmelen.** Object modeling and user interface design. *Addison-Wesley*, 2001.

[9] **M. Lif, E. Olsson, J. Gulliksen, B. Sandblad.** Workspace Enhance Efficiency – Theories, Concepts and a Case Study. *Information technology and people*, 30(4), 2000.

[10] **R. Barker, R. Longman.** Case Method, function and process modelling. *Addison-Wesley*, 1992.

[11] **R. Barker.** Case Method, Tasks and delicerables. *Addison-Wesley*, 1989.

[12] Integration Definition For Function Modeling (Idef0). *Draft Federal Information Processing Standards Publication* 183, 1993 *December* 21.

[13] **R.J. Mayer, J.H. Crump, R. Fernandes, A. Keen, M.K. Painter.** Information integration for concurrent engineering (ICEE) compendium of method report, 1995. [*checked*: 11.09.2004]. *Web address: http://www.idef.com/Downloads/pdf/compendium.pdf*.

[14] **R. Butleris, R. Butkienė, T. Danikauskas.** Business modelling for elicitation of information requirements. Business operation and its legal environment: processes, tendencies and results. *Proceedings of International Conference, Riga: Turiba*, 2002, 67-73.

[15] **R. Butkiene, A. Jasiukevicius, V. Sakys.** The specification of structure of information flows. *Information Sciences, Vol.*24, *Vilniaus university publishing. Vilnius*, 2003, (*in Lithuanian*), 117-124.

[16] **B. Wangler.** Contributions to Functional Requirements Modelling. Doctoral Thesis. *Stockholm University, Royal Institute of Technology, DSV, Akademitryck AB, Edsbruk*, 1993.

[17] **J.T. Hackos, J.C. Redish.** User and task analysis for interface design. *Wiley*, 1998.

[18] **M. Snoeck, G. Dedene, M. Verhelst, A.M. Depuydt.** Object-Oriented Enterprise Modelling with MERODE. *Leuven University Press, Belgium*, 1999.

[19] **S. Robertson, J. Robertson.** Mastering the Requirements Process. *New York, AddisonWesley*, 1999.

[20] **D. Coleman, P. Arnold, S. Bodoff, C. Dollin, H. Gilchrist, F. Hayes, P. Jeremaes.** Object-Oriented Development. *The FUSION Method. Englewood Cliffs, N.J.: Prentice* Hall, 1994.

[21] **M. Kolp, M. Giorgini, M. Mylopoulos.** Organizational Patterns for Early Requirements Analysis. *CAiSE* 2003, *LNCS* 2681, *Springer-Verlag Berlin Heidelberg*, 2003, 617–632.

[22] **P. Coad, E. Lefebvre, J. De Luca.** Java Modelling in Color with UML: Enterprise Components and process. *Yourdon Press, Prentice Hall*, 1999.

[23] **R. Butleris, T. Danikauskas.** Conceptual Data model Design Using Functional Requirements Specification Method. *Proceedings of EMMSAD*'04 *workshop at CaiSE*'04 *conference, Riga, Latvija*, 2004, 221-232.

[24] **A. Aleksandravicienė, R. Butleris, T. Danikauskas.** Data modelling on basis of data flow specification. *Information technologies* '2004, *Conference papers, Technologija, Kaunas*, 2004, (in Lithuanian), 473-479.

[25] **E. Arisholm.** Incorporating Rapid User Interface Prototyping in Object-Oriented Analysis and Design with Genova. The Eighth Nordic Workshop on Programming *Environment Research, Electronic proceedings: NWPER* ' 98, *Sweden*, 1998.

[26] **P.J. Molina.** A Review to Model-Based User Interface Development Technology. *IUI/CADUI* 2004 *workshop, Electronic proceedings, Portugal, Madeira*, 2004.

Received March 2006.

DOI: 10.5755/j01.itc.35.2.12044