Adaptation of a task analysis methodology to the design of a decision support system

Sandrine Balbo and Craig Lindley

CSIRO, Mathematical and Information Sciences

Locked Bag 17, North-Ryde NSW 2113

AUSTRALIA

Emails: [Sandrine.Balbo|Craig.Lindley]@cmis.csiro.au

ABSTRACT The aim of this article is to describe the use of an analytical method for task analysis (MAD) within an industrial development project, APLCRATES. The method has previously been used within research contexts, and has been limited to task analysis and user interface design. Within the APLCRATES project, we not only used the method for defining the user tasks and interfaces, but we also used it for knowledge acquisition and as a powerful communication tool between the different development teams addressing decision support system development, database definition, on-line help, user manual definition, and for communication with the clients. We explain briefly the aim of the APLCRATES project and what MAD is, and then describe in detail how MAD was used and modified to meet the needs of the project.

KEYWORDS

Usability engineering, task analysis, decision support systems, user requirement specification, user participation.

1. INTRODUCTION

CSIRO Mathematical and Information Sciences (CMIS), in collaboration with Integrated Systems Solutions Corporation (ISSC), has developed the Expert Module of a spares assessment expert system (called APLCRATES) for the Manager Codification, Cataloguing and Allowances (MCCA) department of the Royal Australian Navy (RAN).

Activities of the CMIS team included knowledge acquisition, the design and development of the core decision support functions, and the design of the graphical user interfaces (GUI) of the decision support system.

ISSC provided the overall management of the project, as well as the database development (eg. definition, creation, population, report generation functions, import and export functions), choosing and organising the hardware, developing the on-line help information, development of an integrated electronic book, and the development of a tutorial to learn how to use the system and its key concepts.

A key feature of this project was the application by the CMIS team of the MAD methodology (*Méthode Analytique de Description*, or in English: Analytical Description Method). MAD had previously been used as a user task modelling methodology in research

environments. In this development project, MAD was used for modelling user tasks as a framework for interface and general Expert Module application development, as a high-level knowledge model within which detailed knowledge models were developed, and as a communication mechanism between the diverse groups collaborating on project development.

The objective of this paper is not to present a comparative study of task analysis methods, but to show how the MAD task analysis method has been transferred from a pure research environment and successfully applied in an industrial development project and for a type of application, namely a decision support system, for which it was not originally designed.

The paper presents the RAN problem, the MAD methodology, and describes the use of MAD for system requirement specification leading to the development of the APLCRATES User Requirement Specifications (URS).

2. MCCA REQUIREMENTS

The primary task of the Navy MCCA department is the creation of Assembly Part Lists (APLs). Each APL decomposes an assembly (from a simple chair to more sophisticated pieces of equipment embedded on a ship) into the parts it is composed of. Each part within an APL is associated with assessment codes that provide the necessary information for computing the quantity of the part that needs to be stored on board a ship prior to its mission. Decision-making involves a tradeoff between having sufficient spares carried on board ships for them to remain operational, and reducing the total inventory costs involved in the acquisition and storage of parts.

To manage the creation of APLs, five different categories of staff are involved: the system administrator, MCCA managers, MCCA technical staff, MCCA non-technical staff, and trainees. Briefly, the system manager is in charge of managing the database, the managers make sure that the decisions taken are correct, the technical staff associate assessment codes with parts and build the APLs, while non-technical staff conduct data entry and research in the database.

The old MCCA system used for the process of APL creation provides only a character-based user interface. It has many limitations, constituting requirements that a new system must address. The main limitations of the old system are:

- it provides very inefficient access to the database: the end-user has to know the identifier of an APL to access it,
- no download facilities exist from or to the database, ie. to copy an APL from another database, the end-user works from a listing and must retype the complete APL in order to enter it into the new database,
- to exit the system, the end-user has to go back to the main menu,
- learning how to use the system takes a long time, mainly due to its complex methods for navigation, and the use of function keys and shortcut keys,
- very little explanation or help is provided,
- repetitive information must be entered to satisfy the system.

In addition to redressing these deficiencies, the new system must:

- reduce the training time; since it takes approximately 6 months of on-the-job training before a new staff member becomes effective. This problem is compounded by short posting cycles in military forces, resulting in a high staff turnover rate for particular jobs and roles.
- encourage higher quality and more standardised assessment. Trained operators are proficient, but do not constitute "experts" in assessment; the new system is required to take a standardised approach to assessment representing the best of current expertise, and make this accessible to all operators.

These functional requirements for reduced training time and standardised decision making result in the following technical software requirements:

- to automate decisions whenever possible,
- to provide computer supported copy/paste functions between the APLCRATES database and other databases when possible,
- a strong help facility, relevant to the context and available from any stage of the assessment process,

- strong guidance in the decision-making process: asking questions rather than directing the choices,
- improved search facilities.

These requirements meant that it was necessary for the APLCRATES development team to have a good understanding of APL creation and definition tasks, and the requirements pertaining to those tasks from the viewpoint of managers and all other RAN staff involved in the process of APL creation and maintenance. The application experts themselves did not have a single consistent understanding of how assessment should be carried out. A major aspect of the analysis phase of the APLCRATES project was the development of a common and agreed model of assessment expertise. This was done by CMIS human-computer interaction specialists and knowledge engineers working together with expert RAN and Defence Department staff to define a common model.

The APLCRATES project schedule allowed two calendar months to develop the first prototype and complete the User Requirement Specification (URS) for the system. It is during this analysis phase that MAD was most heavily used, providing a systematic foundation for subsequent system design and implementation, including detailed knowledge engineering activities.

3. DIFFERENT VIEWS AND SINGLE TOOL

During the analysis phase at CMIS, we performed two different tasks:

- a complete task analysis to define what the GUI should provide, and how it should be organised. The resulting models provided the basis for screen design, specification of detailed functions associated with each screen, and a high level modularisation of task knowledge as a framework within which detailed knowledge acquisition could occur.
- detailed knowledge acquisition to design the decision support system, resulting in rules, decision trees, and decision tables that could be embedded throughout the extensive interfaces of the system to provide specific decision support for subtasks.

The common basis for these two tasks was a series of interviews with the users; the interviews were performed by the knowledge engineers (who were primarily in charge of knowledge acquisition) and the human-computer interaction specialists (who were primarily in charge of the design of the GUI). Even if the overlap between the two tasks is extensive, very different formalisms were used to transcribe on to paper the understanding obtained from these interviews. The knowledge engineers used flow charts, decision tables, and rules, while the human-computer interaction specialists used MAD.

We will discuss very briefly the technique initially used by the knowledge engineers; in practice the whole team very quickly shifted to the use of MAD. To find a complete discussion from the knowledge acquisition point of view, refer to [Lee 97].

For knowledge acquisition, flow charts were used to get the overall organisation of APL creation and definition tasks. The limits of effectiveness of flow charts were quickly reached, particularly the fact that they can only represent strict sequential activities, while the work performed at the MCCA department is not simply sequential, but also involves a lot of interleaving tasks. Building an APL can take several months, and is certainly not done in one uninterrupted sequence.

For the task analysis, MAD was used. MAD provides a methodology to perform task analysis, following a hierarchical decomposition of the tasks; MAD is described in detail in the next paragraph. The human-computer interaction specialists used MAD because the method is well documented, has been proved to work in many different research contexts, and because it is easy to use (as demonstrated in the next sections below). Unfortunately, there was no time to perform a comparative study among different task analysis methodologies before the start of the project.

A few weeks after the beginning of the project, as the knowledge engineers reached the limitations of flow charts, we decided to adopt the MAD formalism as a common formalism within the CMIS team. As explained in section 5.2, the MAD formalism not only was used as the basis for both knowledge acquisition and GUI design, but also by the ISSC team, and as a communication tool with the clients. A brief

description of the MAD methodology is provided in the next section. Use of the MAD formalism in the context of the project is explained in section 5.

4. THE MAD METHODOLOGY

MAD provides a methodology to gather information on the operational tasks performed by users, based upon extensive interviews and leading to the formalisation of task descriptions. The interviews with the end-users are semi-directed by the MAD specialists. In the first part of the interview, the end-user will not be interrupted and will be asked to describe how s/he is working. This allows the analysts to gather the user's representation of their work. The second part of the interview is more closely directed: the "Why and How" technique is used to split the end user's goals from the methods they follow to achieve these goals [Sebillotte 95]. This second part of the interview process enables the completion of high level task representation using the MAD formalism. "The MAD formalism offers an extension of the key-concept of hierarchy (in the sense of structural parsing) of abstraction levels, using logic-and-time graphs of objects, from the most general (root) to the most specific (task leaves, ie. elementary tasks). In addition, an object-based structure has been chosen to offer both declarative and procedural statements; each object is an independent entity, which corresponds to the requirement of representing conceptual entities significant to the user" [Scapin 90]. The concept of a task is represented using this object-based structure. Each object is characterised by the following [Sebillotte 95]:

- the name: a short and clear description of the task under consideration.
- the goal: states the objective achieved by executing the task.
- the initial state: a list of objects representing the part of the world in which the task will be carried out, and that might be modified by the execution of the task.
- the preconditions: a set of predicates expressing constraints upon the objects of the initial state.

- the body of the task: expresses how the task is executed. It can be either an elementary or a composite task:
 - * an elementary task is generally represented by a verb that expresses the action that it refers to,
 - * a composite task can be decomposed further into a set of sub-level tasks (that will be either elementary or more composite tasks) linked together by one of the following constructs:
 - SEQ (sequential): subtasks are executed one after the other,
 - PAR (parallel): no constraint on the execution order of the subtasks,
 - SIM (simultaneous): true parallelism in the execution of the subtasks,
 - ALT (alternative): only one of the subtasks needs to be executed.
- the postconditions: are a set of predicates expressing constraints upon the objects of the final state.
- the final state: a list of objects representing the part of the world in which the task was carried out, and that was modified or created by the execution of the task.
- The task attribute: a characteristic of the task and may be any of the following:
 - * @ (iterative): the task may be repeated, several times,
 - * OPT (optional): the task does not necessarily have to be carried out,
 - * PRIOR (priority): the task can interrupt other tasks,
 - * INT (interruptible): the task is or is not interruptible.

5. ADAPTATION AND USE OF MAD

A few modifications to the MAD formalism were necessary while characterising each task during the interviews with the MCCA managers and staff (ie. when building the tree and associating with each task a name, goal, pre- and post-conditions, etc.). We explain below what these modifications are and why they were necessary, before presenting what we used

Figure 1: Example of use of the MAD formalism

the MAD formalism for, within the analysis phase of the project. Figure 1 and table 1 illustrate our discussion. Figure 1 presents an example of how the MAD formalism was used within the project. Table 1 summarises MAD usage and the different actors involved in APLCRATES development. Four actors are distinguished: the CMIS development team, the ISSC development team, the MCCA managers and the MCCA staff (technical, non-technical, system administrator and trainee).

5.1. MAD modifications

The three main modifications were:

- the dropping of pre- and post-conditions,
- the addition of a comment slot to group information specific to knowledge acquisition,
- the addition of the source of information.

The first and most significant modification concerns dropping the initial state, precondition, postcondition and final state. Dropping the precondition follows the addition, in the tree, of some conditions (for example in figure 1: 'if APL exists in

Navallog' prior to the 'Import APL' task) to be realised prior to the execution of a task. It seemed more natural and easy to formulate conditions that way, instead of using objects to be defined in the initial state, and then used in a postcondition. We did not need the definition of these objects for our initial goal to acquire a general understanding of the tasks performed by the MCCA staff. The same applies to the postconditions and final state.

For the knowledge engineers, the initial MAD formalisation was very useful for organising the information as it was gathered. Since the MAD formalism did not provide a structure, or a slot, in which to place this information, we created one by adding a comment slot to each task (similar to the name or goal slots). The comment slot allows us to associate information with the appropriate task in a structured way. Information used in the comment slots included algorithms to be applied, databases accessed, default values, field length and so on.

The third important modification done to the MAD formalism was the addition of another slot for specification of the source of information. The source

of information clarifies whether the task is performed on paper or on the computer, when it is carried out with the help of the computer, and whether the end-user or the computer is performing the task. This source of information was represented in the tree by a plain rectangle box for tasks performed by the end-user within the system (for example, in figure 1, entering the 'Project title' in the system), a round box for automatic tasks (for example, in figure 1, the computation by the system of the serial number, depending on the SMG code entered by the end-user), and a dashed-line rectangle box for manual tasks (for example, in figure 1, the opening of a folder including various papers concerning the project being created).

A few minor modifications were done as well, concerning the use of constructors and attributes.

For tree constructors, we only used SEQ, PAR and ALT. We did not need the SIM constructor since there is no true parallelism in the procedures followed for constructing and managing APLs (to control database consistency, the same data cannot be modified by two people at the same time).

The only task attribute we used was the iterative (symbolised by @). We did not use the optional attribute, but added the mandatory one to some elementary tasks, essentially when it concerned entry of mandatory database fields (for example, in figure 1, the field 'SMG code'). The non mandatory elementary tasks are not optional either, they have to be performed at one stage for the high-level task to be completed, but do not need to be performed to start the other tasks. In the example, to start the 'APL development' task, only the mandatory fields of the 'Define Project information' must be completed. The notation could be extended to deal with cases like this

by incorporating separate symbols to distinguish mandatory but non-sequential subtasks.

5.2. Use of MAD

We used MAD in many different ways; for task analysis, knowledge acquisition, the design of both the decision support system and the help/tutorial related items, and as a communication tool among all actors involved in the APLCARTES project.

First of all, MAD provided us with a reliable way of organising our understanding of the MCCA task. While performing the first rounds of interviews, we also used MAD as a tool for presenting the results of the analysis to the MCCA managers and staff to demonstrate we understood their tasks: both the hierarchical decomposition of tasks and the components of acquired knowledge. In this way MAD not only helped us to make sure that our understanding of the task was correct, but also helped to fix the vocabulary used subsequently within the APLCRATES system.

Having constructed the MAD trees for the MCCA tasks, the design of the GUI for the decision support system was made easy. There is no doubt within the development team that the understanding we had of the organisation was reflected through the user interfaces with the help of the MAD descriptions. The MAD trees indicated groups of items that had to be kept together within the GUI, as well as ones that could be split through different sets of screens; for example, from figure 1, all information under the 'Define Project information' task block is presented in one screen. These trees also provided the basis for the organisation of menus and buttons, and, more importantly, they provided the general map for navigation through the different screens. The generic

Actors involved	MAD usages
CMIS, MCCA managers & staff	structuring our understanding of the task
CMIS, ISSC, MCCA managers	fixing vocabulary
CMIS, MCCA managers	structuring knowledge acquisition
CMIS, MCCA managers & staff	design of the GUI
ISSC, MCCA managers	defining topics to be covered within the help system and the tutorial
CMIS, ISSC, MCCA managers	writing the URS

Table 1: Summary of the MAD usages

nature of this process for going from the task description to the GUI design would be worth investigating.

In fact, working on the MAD formalism of the MCCA task provided a very good understanding of the work environment. In addition, this representation provided a constant communication loop between the development teams and the MCCA managers and staff for validation purpose.

The use of MAD was so well accepted by all actors of the APLCRATES project that the models expressed by the MAD formalism were used to compose the heart of the User Requirement Specifications for the project.

6. CONCLUSION

The APLCRATES system answers the users' needs and follows the work organisation that was successfully understood by the development team through the MAD description. This successful understanding critically relied upon strong collaboration with the end-users. The first prototype has been intensively tested by the human-computer interaction specialists (for usability) and by the MCCA managers (for the correctness and completeness of business rules), and the major changes that came out of these tests were limited to the organisation of the decision making process. Today, the implementation of the first version of the product is in use by MCCA staff and the only complaint received so far concerns some slow access to the database in the RAN busy networking environment. We are planning shortly to conduct some more usability tests on the RAN site.

The fact that MAD was successfully used for high-level knowledge formalisation opens the possibility of applying task models to new domains such as knowledge-based systems. Some of the modifications that were brought to MAD in the context of this project must be seen and understood in this light.

It may be noted that using the MAD description helped us gain an understanding of MCCA tasks and

their decomposition, but the context of the work itself was only understood later on in the process. In fact, the MAD methodology assumes that the work context is already well understood. To understand the context of the work, some other tools like the contextual design methodology developed by [Holtzblatt 96] could be used, and this would complement MAD to provide a comprehensive analysis methodology.

ACKNOWLEDGEMENT

Thanks to the various reviewers of this paper for their helpful comments and to all members of the APLCRATES project (development team, MCCA managers and end-users) for a dynamic working environment.

REFERENCES

- [Holtzblatt 96] Karen Holtzblatt & Hugh Beyer, "Contextual design: principles and practice", in *Field methods for Software and systems design*, John Wiley & Sons, 1996.
- [Scapin 90] Dominique Scapin & Christine Pierret-Golbreich, "Toward a method for task description: MAD", in *Work with display units 89*, L. Berlinquet & D. Berthelette (Eds), Published in Elsevier Science Publishers B.V. (North-Holland) 1990.
- [Sebillotte 95] Suzanne Sebillotte, "Methodology guide to task analysis with the goal of extracting relevant characteristics for interfaces", Technical report for Esprit project P6593, INRIA, Rocquencourt (France), April 1995.
- [Lee 97] Maria Lee, Sandrine Balbo, Craig Lindley & Mikhail Prokopenko, "Combining Rule-Based Knowledge Acquisition with Task-Oriented User Analysis for Dispersed, Embedded Expert Systems", PACES'97 conference proceedings.