# Student task modeling in design and evaluation of open problem-solving environments

**N.K. Tselios[1], N.M. Avouris[1*], M. Kordaki[2]**

[1]University of Patras, ECE Department, HCI Group, GR-26500 Rio Patras, Greece

[2] University of Patras, CEI Department, GR-26500 Rio Patras, Greece

**nitse@ee.upatras.gr, N.Avouris@ee.upatras.gr, Kordaki@cti.gr**

Corresponding author: Nikolaos Avouris

University of Patras, ECE Department,

HCI Group, GR-26500 Rio Patras, Greece

tel +30-61-997349  fax (30) 61-997316

N.Avouris@ee.upatras.gr

# Student task modeling in design and evaluation of open problem-solving environments

## Abstract

Design and evaluation of computer-based open problem solving environments is a non-trivial task. Definition of a design framework, which involves a strong field-evaluation phase, has been the subject of the research described in this paper. This framework is based on the concept of student task modeling. Tools to support design and evaluation have been built and used in the frame of this study. The framework and the developed tools have produced promising results during the evaluation of an open problem-solving educational environment.

**Keywords** Student task model, Task Analysis, User interface design, Usability evaluation, Open problem solving educational environments.

## 1.Introduction

One class of computer-based learning environments that has drawn the attention of the research community lately is that of *open problem- solving educational environments*. These are typically interactive systems that support solving of classes of problems by the users with emphasis in the active, subjective and constructive character of learning (von Glasersfeld, 1987). In these environments the student’s activity is not reduced to a sequence of pre-defined tasks combined with an evaluation system of the correct answers, as in many traditional learning environments. Instead a set of tools is provided, through the use of which the user can construct solutions to an open set of problems. These environments provide a context to let users actively explore certain abstract concepts. Typically such environments are characterized as *microworlds* (Papert, 1980, Balacheff & Kaput, 1996). Open problem-solving educational environments provide immediate intrinsic feedback to the user's actions in a very simple, direct, visual way (Laborde & Strasser, 1990), providing opportunities for exploration of different, new and linked representations of the same concept (Kaput, 1987).

Design of such environments is not an easy task. While there seems to be wider acceptance of the importance of these environments in learning and problem solving, at the same time there is a growing concern relating to the lack of methodologies and tools supporting their design and evaluation (Kordaki and Avouris, 2001). Some first observations on user interaction with these systems recognize the exploratory nature of the process, through which the user can advance the knowledge of the system and the tools concerned. The design may also involve study of possible alternative user strategies, facing typical tasks and alternative ways in which the users could interact with the provided tools. However this user behavior is not fully anticipated. Students can approach problems in different ways and can make mistakes in the process. From the constructivist perspective, it is exactly these mistakes that can be treated as opportunities to learn.

A frequently observed design approach incorporates interaction design techniques inspired by the Human Computer Interaction (HCI)[*] area of research, which primarily concerns users engaged in typical workplace environments (Inkpen, 1997). However the design according to this approach, often needs enrichment with observational data generated during field studies, thus necessitating an iterative, user-centered development process.

[*] For a list of abbreviations look at the end of the paper

One issue worth investigation is the applicability of well-established *task modelling* techniques in this context. Task and goal modelling have been extensively used in HCI research and practice in order to build representations of users understanding, knowledge, goals and task execution. The proposed models, based on cognitive science theoretical and experimental work, have already been applied both during *design* and *requirements capturing* as well as during *user interface evaluation*, as discussed in section 3 of the paper. However, typically these techniques attempt to model execution of routine tasks by ideal users, offering a mechanistic view of interaction especially for prediction purposes, instead of analyzing actual users' behavior.

The main objectives of this study are: first, to define and apply a new design and evaluation framework for open problem-solving educational environments, based on task modeling techniques and second, to present appropriate tools which have been developed to support the design and evaluation process.

The proposed framework involves incremental application of task modeling techniques, starting from an expected ideal students' task model defined during design, subsequently comparing the observed student behavior to this original model during field evaluation. The first findings of this proposed methodology and tools, as derived from the evaluation of an open problem-solving environment, are very promising.

This paper is organized in the following way: an overview of issues related with the design and evaluation of educational environments is presented in section 2, followed by a review of existing task design and evaluation techniques. In section 3, the proposed design and evaluation framework is presented. In section 4, a description of the tools that have been developed in order to support this framework is provided: the *Usability Analyzer Tool* (UATool) that supports the field evaluation phase and the *Cognitive Modeling Tool* (CMTool), that facilitates the task modeling and evaluation phase. In section 5 a case study of application of the proposed framework in the design and evaluation of a learning environment regarding geometrical concepts is reported. In section 6, a comparison of the findings of the study to an existing evaluation technique applied to this environment is included. Some conclusions on the advantages of the proposed technique and its applicability in evaluation of open problem-solving educational systems are also included at the end of the paper.

## 2. On usability evaluation of educational software

Currently, various evaluation methods are used to assess usability of educational software. These methods have been inspired by research in the Human-Computer Interaction field and can be classified into three main categories:
*i) Testing methods,* like the *thinking aloud protocol*, *Co-discovery* and *performance measurement*, (*ii) Inspection methods, like heuristic evaluation* based on usability principles - the heuristics, *Cognitive walkthrough*, *Guideline checklists* used often as a basis for comparative evaluation and (*iii) Inquiry methods,* like interviews of the students or evaluation questionnaires. For a discussion and a comparative presentation of use of many of these methods, see Avouris et al, (2001) and Tselios et al. (2001b).

The effectiveness of the above methods on various kinds of educational software has been however questioned. Squires and Preece (1999), doubt about the applicability of traditional predictive evaluation techniques, especially *checklists* because these do not encompass a consideration of learning issues. They also mention that most forms of *walkthroughs* are strongly cognitively oriented and require detailed knowledge, so they are not suitable for educational systems, which encourage creativity and can be used in different ways by different students.

Squires and Preece (1999), also argue in favor of a convergence of methods of research of educational software and human computer interaction. In particular they propose an extension of heuristic evaluation by taking into account the socio-constructivist learning perspective. In addition, Mayes and Fowler (1999) argue that educational software design should be focused on design of effective tasks rather than interfaces, since learning cannot be approached as a conventional task. Therefore, they stress the importance of evaluating suitability of tasks for their educational purpose. Sedig et al. (2001) also suggest that educational software is effective when knowledge is implicitly embedded in the task. So, task modeling methods seem to be appropriate to be used in evaluation of environments of such characteristics.

An alternative approach to the traditional usability evaluation methods, has emerged as a need from the above discussion. One alternative is the "*Model-based evaluation*" approach, proposed during the last years in the HCI field. According to this approach, the evaluation can be based on observation of typical users during task execution and subsequently modeling of their behaviour. The study of interaction between the user and the provided tools in the microworld is performed in the frame of goal-oriented activity during typical task execution and is related to the expected task patterns as anticipated by the system designer. This has been the approach proposed in this paper. This approach is complementary to other usability evaluation methods and can be proved suitable to open educational software evaluation, as it can relate problem solving strategies to interaction and tool design. Before we proceed with the proposed model-based evaluation framework, in section 3, we provide a brief survey of existing task modelling techniques in the next section.

## 2.1 Task analysis design and evaluation techniques

The proposed framework is based on a *Task modeling and Task Analysis* (TA) approach. The objective of TA from the HCI perspective is to facilitate understanding the user's tasks and goals by explicitly describing what the user's tasks involve (Kieras, 1996). There is already some evidence that task analysis may substantially contribute towards the successful design and evaluation of user interfaces (Lansdale and Ormerod (1994), Richardson et al. (1998)). A number of successful case studies have been reported in a variety of design projects (Olson and Olson (1990), Gong and Kieras (1994), John and Wayne (1994), Haunold and Kuhn (1994), Ainsworth and Pendlebury (1995), Umbers and Reiersen (1995), Paterno and Mancini (2000), Patterno' and Ballardin (2000)).

By using a task modeling technique the software design could be described more formally, analyzed in terms of usability and can be better communicated to people other than the analysts (Johnson and Johnson (1991), Abowd (1992), Richardson et. al (1998). Wilson et al. (1993) also propose such models to be used as a core for task-centered design of user interfaces. As a result, the HCI community has currently focused on investigation of underlying supporting theories and further development of task analysis methods and techniques based either on empirical data or on psychological models (Card et al. 1983). Some of these techniques are: ATOM, (Walsh, 1989), CTA (Barnard, 1987), HTA (Shepherd, 1989), CLG (Moran, 1981), GOMS (Card et al., 1983), NGOMSL (John and Kieras, 1996), TKS, (Johnson and Johnson,1991) and TAKD (Diaper,1989).

Task analysis has also received some criticism, relating to its applicability, the lack of a detailed methodology for applying specific TA methods into the design cycle, and the large effort required for applying it. Some concerns are also related to the lack of tools supporting the proposed techniques. John and Kieras (1996a) argue in favor of use of cognitive modeling for usability evaluation, based on the sound theoretical foundations of the technique and promising results of specific case studies. Additionally, Lim (1996) describes how task analysis is incorporated explicitly throughout the design cycle in the MUSE method for usability engineering.

The techniques mentioned here often use different notations, however they share a common foundation and key assumptions, the most important of which is that they attempt to model ideal user performance. Additionally, in most cases these techniques are used when the task structure of the typical system use is well defined, as happens in command line interfaces. So, in cases where the interaction is not a trivial sequence of actions, as in open problem solving educational environments, the suitability of these techniques is questioned. Moreover, there are application areas, like learning environments, in which flaws in user performance are often the rule and through them the user builds new knowledge. Designs that do not cope with such “deviations” can present users with considerable difficulties. In addition, traditional task modeling approaches, have been mostly used to determine usability of a goal oriented process in terms of execution time, frequency of errors, efficiency and effectiveness. These approaches need however to be redefined and extended to educational systems, as they are not always suitable. Despite these concerns, the wide acceptability of these techniques led us investigate applicability of Task analysis (TA) in the design and evaluation of open problem solving environments.

## 3. The proposed framework for evaluation and design

In this section a methodological framework for the design and evaluation of open problem solving environments is described. According to this framework, *software prototypes* of the educational environment are developed, based on design *specifications* and assumptions on expected typical problem solving strategies. These prototypes are subsequently revised and adapted following a *field evaluation* phase. The revision of the design specifications during the field evaluation study is a *model-driven process*. The most important aspect of this framework is the iterative development of a *task model* by the designer during the first phase, subsequently modified by taking in consideration user behavior during field evaluation. In order to develop this model, typical users' interaction within the problem-solving environment is closely monitored and observed. The method of task analysis adopted here is the one of *Hierarchical Task Analysis*, with some modifications, as proposed by Shepherd (1989). It should be mentioned that our intention has been to build a conceptual model, referring to the way the student views the system and the tasks in accomplishing certain problem solving goals. HTA bears many similarities to the goal-oriented cognitive modeling techniques such as GOMS (Card et al. 1993). However, the purpose of our analysis is to reflect on the observable behavior of users and to identify bottlenecks on the human-system dialogues rather than to understand the internal cognitive processes as one performs a task. By using this approach, special attention is paid to specific tasks involving problem solving activity, such as applying previous knowledge on new tools and objects in order to face novel challenges.

The proposed framework comprises three phases (Figure 1): i) Preliminary analysis and development of a model of the anticipated task execution by users, called Designer Task Model (DTM). ii) Field evaluation, observational data acquisition and data interpretation and analysis. iii) Development of the Student task model (STM), based on field data. Modification of the DTM, developed in (i),  according to the developed STM. Identification of usability problems, Generation of new design specifications. These three phases are discussed in more detail in the following.

Figure 1 Outline of the evaluation framework.

## 3.1 Phase (i): Preliminary analysis and original designer's model (DTM).

During this phase the designer, based on theories of learning, domain knowledge, requirements analysis, the expected user's characteristics and the social environment in which the system will operate, defines the primary designer's task model (DTM). Part of this model is the *Designer's View on the typical Student Task Model,* which represents the way the designer expects the student would interact with the problem solving environment in order to accomplish typical tasks. It is assumed that this model is made explicit using the *Cognitive Modeling Tool* (*CMTool*) discussed in section 4. The tool is used to manipulate a graphical representation of the model (see Figure 2). This model contains the expected high level *goals* of the student, further decomposed into sub-goals. Generally, in an open problem-solving environment the individual students can approach a given problem in many ways, thus expressing their individual differences. These may be reflected in the designer's task model, which can contain many alternative goal hierarchies. This model at the lower sub-task level contains tools-related activity. So it can be used to design these tools effectively in order to support typical expected task execution. The use of the task model during this design phase is the traditional task modeling activity as discussed in section 2.1.

## 3.2 Phase (ii): Field evaluation study.

Prototype field evaluation is the phase of the proposed framework during which the designer effectively receives feedback from typical users of the software in order to adapt the original design, supporting the observed typical users needs. The original hypotheses about the way users are anticipated to interact with the system may be revised and extended. According to (Orhun, 1995) effective communication between the user and the designer and the user and the devices require mechanisms that allow adaptation of the original assumptions. The explicit representation of the designer task model (DTM) created in the previous phase (i) is essential in order to devise and support this adaptation process. In our case this *field evaluation* process serves to validate original designer's assumptions against actual student typical task execution, through annotation of discrepancies between DTM and STM.

This field study may take place in a usability laboratory or in a school environment. Participants of the experiments must be representative students with same characteristics as the expected users of the software. The students receive a list of representative tasks, in terms of system's proposed functionality, to accomplish. Their behaviour should be closely observed and log files of interactions and videos can be useful material for further analysis. By observing the individual student's interaction at the keystroke level (e.g. through log files) the users' models reflecting their conception of the system is constructed. Students' views about missing functionality of the systems can be also detected.

For each student, $n=t*s$ models need to be built, where $t$=number of high level tasks undertaken and $s$ is the number of different strategies used to solve a given tasks. The *Usability analyzer* tool (UATool) , discussed in section 4, supports the keystroke level monitoring and analysis process.

### 3.3 Phase (iii): Student interaction modeling and DTM refinement.

Students' deviations from the expected behavior are monitored at the keystroke level and are mapped at the task level during this phase. If a usability-related mistake is detected then the task where this unexpected interaction occurred is specially annotated with a text description and a grading of the severity of the problem. If an unanticipated problem solving strategy is observed, this is added incrementally to the original designer model. Remarks on additional functionality required to support these unforeseen strategies are also made. Conceptually wrong problem solving strategies can also be included in the student's interaction model, since they are often useful for educational purposes or can be used as typical misconceptions to be tackled by the system.

Following this process, a combined *Student Task Model* (STM) is derived which contains all the students' solution strategies detected and analyzed. The purpose of this process is to unveil new ways of interaction that the original designer model could not support and determine the gap between the designer and the student models. After this analysis the original *DTM* is revised and augmented. A number of requirements concerning re-design of the microworld, can be derived by this revised DTM.

## 4. Tools to support the framework

The effort required to construct a task model and to maintain various instances of it is one of the main reasons discouraging the wider adoption of task modeling (Kieras, 1996). This finding inspired us to develop appropriate tools to support the previous described design and evaluation framework.

### 4.1 The Cognitive Modeling Tool

***General characteristics of CMTool*** The *Cognitive Modeling Tool (* CMTool[1] ) has been developed to facilitate the task modeling process. It has similarities with other Task Modeling tools, like *Concurtasktrees* Tool (Paterno et. al 2000) and *Euterpe* for GroupwareTaskAnalysis (Van Welie et al. 1998). As CMTool is specifically designed for the evaluation of open educational environments it has many innovative features, that distinguish it from other tools of this kind, as described in this section. Task models are structured in a hierarchical way, as shown in figure 2. CMTool is expected to be used by non-HCI experts, so it is based on an intuitive direct manipulation approach for editing and modifying task hierarchies. Any specific node represents a *task* relating to a student's

[1] CMTool has been developed for the Windows environment using Visual Basic 6 and Win32 API calls and is freely available for non-commercial use.

*goal*. The sub-tasks which serve to accomplish this goal are associated to the node. The defined *plans* are inserted as text entries and are not processed by the tool for syntactic correctness, see Appendix for plan annotation symbols. Additional information and *comments* on the goal accomplishment can be attached to the node, as shown in task 1.2.2.1 of figure 5 (properties box). Plans representation is different than other tools, like *ConcurTaskTrees,* where the task relations are inserted in the task graph and make part of the task structure. However, in the *CMTool* case the main concern was the simplicity in description and inspectability of the tasks structure, which is achieved through the selected representation.

Moreover, each task node can be associated to specific *tools* used. By this way problems associated with specific tools could be analyzed. An *student task model* can therefore be annotated with comments relating to task execution. A special notation has been defined, included in the Appendix, extending the HTA plan notation (Shepherd, 1989), with tokens referring to user deviation from expected task execution, as shown in plan 1.1 of Figure 4.

The above features of CMTool are not found in other task modeling tools. This is because the framework that CMTool supports involves field studies, so the possibility is provided of associating observational comments to task-related description of activities. This is one of the most powerful and innovative aspects of the CMTool environment.

*Fig 2. Cognitive Modeling Tool : The main task model building space. Part of the designer task model(DTM) of the task discussed in section 4 is shown*

### ***Main CMTool functionality***

- The original keystroke log files and the corresponding task model can be shown concurrently in CMTool, as shown in Figure 2. The possibility of dragging an event of the log file to the goal structure, which results in the introduction of a new node, filled with the action description, facilitates the process of goal structure building.

- A novel and time-saving functionality of the CMTool is its ability to automate synthesis of existing task structures. Using this feature, various sub-goal structures can be combined or temporarily hidden away in a task model, according to the degree of detail required in the context of a particular study.
- The evaluator can select parts of a task structure representing a specific problem solving strategy, which can be stored for future reference or comparison with other users' strategies.
- When the analysis is focused on the time required to accomplish a task, *time related information* can also be stored on each task node and appropriate calculations regarding the time required for a task to be accomplished can be carried out automatically.
- CMTool permits exporting task models in *various alternative representations*: graphical-hierarchical tree view, sequential view and structured report view are produced automatically, all consistent with each other.

***Analysis characteristics of CMTool*** Task models are stored in a database, together with qualifying information (DTM, STM, user id, etc.). Based on this data base, quantitative analysis tools are supported to extract useful statistics related to the analyzed tasks, such as number of keystrokes required to achieve a specific goal or sub-goal, mean time required and interaction complexity of specific user model compared to primary designer’s expectations or to revised and adapted model. Additionally the CMTool supports storage of various users’ characteristics and further analysis is supported, focused on user characteristics such as age, grades and gender.

The possibility of analyzing further the observed system usage according to a number of dimensions permits evaluation of the software environment both in terms of usability and learning. The dimensions of analysis are : (i) problem solving strategies, (ii) individual students, (iii) specific activities to carry out a strategy, (iv) tools used, and (v) types of problems detected during interaction.

This analysis is supported by a *visual query construction environment* shown in Figure 3. The constructed models could be analyzed across any combination of these five dimensions. Through this environment queries can be built concerning for instance identification of all encountered problems related to tool X, identification of all usability problems concerning male students related with problem solving strategy Y, etc. From quantitative data, like frequency of usage of the provided tools, the evaluator using this environment can focus on specific errors or tools. This representation of summative value is missing from the hierarchical task models view. Also educational evaluation of the environment (e.g. strategies selected, tools used, strategies relation to student characteristics etc.) can be performed through this analysis facility of the CMTool.

This capability of CMTool distinguishes it from other Task modeling environments and tools (e.g. Conscurtasktrees), which are not specific to evaluation of educational software and so they lack this analytical power along the described dimensions. Additionally, the CMTool is designed with the objective to be used by educators as a tool for evaluating and analyzing the educational effect of student interaction with the environment and therefore, as presented in this section, it is based on intuitive task description techniques and analysis tools, in contrary to other tools that are mostly used for usability evaluation by HCI experts.

*Figure 3.CMTool:. Visual queries construction environment.*

### 4.2. The Usability Analyzer Tool (UATool).

An additional tool developed to support the proposed framework has been the *Usability Analyzer Tool (UATool*), shown in Figure 4. UATool supports analysis of the keystroke log files and is used in phase (ii) of the framework.

*Fig4.Usability Analyzer: On the left the log file of the user is shown. In the main window the corresponding user screenshots can be studied, at the bottom evaluator's comments are included. The data shown are related to the task discussed in the case study of section 4.*

It comprises a log file parser, which converts difficult to understand keystroke sequences, mouse movements and click streams captured during the field study into human readable information. This is achieved through an association table between the environment functionality and relevant keystrokes. Additionally, screenshots captured during the logging process can be shown, thus extending the traceability of observed interaction. Comments on the interaction activity, made by field observers could also be associated to log-files and screenshots. Moreover, UATool can execute a massive parsing against all the collected log files and store extracted information into a database, such as number of actions, frequencies of specific keystroke sequences, and performance data. The results can be communicated to the user of the tool (e.g. the evaluator) either in the form of graph charts or structured text reports.

## 5. Case study: Design and evaluation of C.AR.ME.

### 5.1 Context of the study

In this section a case study of use of the presented framework and tools is included. The objective of the study was to evaluate and redesign the open problem-solving environment C.AR.ME (Conservation of Area and its Measurement). This is an open problem-solving environment designed to support 12-14 year pupils exploring the concepts of conservation of area and its measurement (Kordaki & Potari, 1998). C.AR.ME. offers the opportunity to explore a variety of geometrical representations of the above concepts, which can be produced in many ways. For instance areas can be measured by selecting and iterating a variety of *units* to cover them. In the case of conservation of area the pupil can split areas in parts and recompose them to produce equivalent areas while tools are provided for dynamic transformation of a polygon to different shapes of equivalent areas. To support the users creating these representations, the environment provides a set of tools. By studying these representations the users of CARME have the opportunity to experience the conservation of area in shapes, which are difficult to produce in the paper and pencil environment as well as to discover their common properties.

*Fig 5. Screenshot of the CARME environment.*

The original task model (DTM) containing many alternative approaches to solve the problem of measuring an area of a geometric shape or transforming into an equivalent one was developed. The evaluation process took place in a school Computer Laboratory. 30 pupils aged 13-14 took part in the experiments. Each one of them interacted individually with the software. Pupils were given two typical “conservation of area and its measurement” tasks. The first task involved transformation of a non convex polygon to another one with equal area and the second task involved comparison of this polygon to a square non easily comparable by ‘eye’. Pupils had to perform both tasks ‘in any possible way’. One of the researchers participated as an observer without intervention in this experiment and recorded the pupils’ solution strategies in field notes. Pupils’ interaction with the environment was recorded in log files and screenshots of their workspaces were also saved for future reference. A typical such screenshot is shown in figure 5.

114 log files containing various strategies, attempted by the 30 pupils were obtained. After the experiment, evaluation of interaction and the learning effectiveness of the environment, took place. According to the analysis framework presented in section 3.2, we chose strategies of five pupils (three male, two female) concerning the task which involved “*transformation of a non convex polygon to another shape with equal area*”. For the second task which involved “*comparison of this polygon to a square*” the solutions of seven pupils, (five male and two female), were analysed. The selection of the solutions to be analysed was based on the richness of the problem solving strategies involved.

Eight (8) and fifteen (15) different problem-solving strategies where analyzed respectively, and modeled with the aid of the CMTool. Task models were inferred from click stream data related to the corresponding screenshots of students working environments. Strategies recording and relative notes of observers helped us to clarify the problem-solving process.

## 5.2 Task-model based Usability Evaluation of CARME

Feedback from task analysis helped us unveil both *syntactic* deviations, identified at lower levels of task hierarchy, mostly interpreted as inappropriate usage of tools due to usability issues, and *semantic* ones. Semantic issues such as unpredicted goal sequences, were contributed either to learning misconceptions on the subject or attempts of the users to set up strategies not foreseen by the original designer model. Next a combined STM was produced as a result of reflection and discussion between the evaluators and designers of CARME.

This combined model helped us identify frequency of tool usage as well as tools with specific problems. For example, a student during interaction with the CARME environment, transformed a polygon to a square with equal area by dividing it into two shapes, measuring the area of each shape and then adding them. Finally, the student attempted to design a square with equal area to the measured polygon's area. So the student had to draw a square of side equal to the square root of the measured area of the polygon. However the *line drawing tool* of CARME did not support drawing a line of a given length. So the student used a trial and error approach, involving drawing a line segment and measuring it, until achieving the desired length. Such an approach, not anticipated in the original DTM model, helped us identify this limitation of the specific tool and request accordingly extension of its functionality.

*Table 1. Frequency of tool usage in transformation task.*

Results from analysis of tool usage are shown in Table 1. Efficient educational software design is expected to promote usage of tools, which embody important domain concepts.

One aspect to be examined was if tools, especially those used frequently as identified from the modeling process, were transparent to the learning task. For instance, the polygon design task is expected to be carried out by three actions, according to the DTM. However, our analysis indicated that 3.75 actions were needed on average to carry out the specific task (Table 1) because of poor usability of the polygon drawing tool. This is an indication that pupils may have paid unnecessary amount of attention and effort to routine tasks instead of reflecting upon domain concepts through fluent manipulations of tools and geometric objects.

Another noticeable result is that task actions related to specific tools seems to follow an exponential distribution with a small amount of specific tools having very high popularity while others receiving little attention.

## 5.3. Detailed Interaction Analysis Example.

To illustrate how the framework has been used in detail, an extract of interaction analysis is presented here. The presented task involves creation of a polygon and generation of geometric equivalent shapes of the same area, as for example squares, rectangles and triangles. In CARME this can be achieved using many tools, like gridlines, measuring units or automatic transformation of a reference shape. In the described example a pupil has opted for this last approach. A short extract of this pupil's interaction with the system is shown in Table 2. The designer model for this task, shown in Figure 2, contains two sub-goals: *(1.1) Design of a reference polygon*, *(1.2) Generation of an equivalent shape using relevant commands*.

*Table 2. Example log file and comments on the interaction behavior.*

The workspace of this particular pupil during this process is shown in Figure 3. The reference shape is polygon (a). Equivalent shapes built subsequently are shown as (b), (c), (d) in Figure 3. Analysis of task execution by this pupil is discussed in the following. Three specific cases of observed deviations of the user task execution in relation to the DTM in Figure 2 are included:

(a) The *Reference polygon drawing* task (1.2.1) has been originally designed as follows: The pupil is expected to draw all the sides of the polygon except the last one and then to select "*end draw*" to complete the polygon. From user actions [1-7] of the log file in Table 2, it is deduced that the pupil attempted to complete the polygon by drawing the last point of the final segment close to the starting point, subsequently selecting the appropriate command “end draw”. This was marked as a low severity syntactic deviation in the corresponding Student task model. The additional task {1.1.3} was included in the task execution plan, as shown in Figure 6. A remark was also made that it is desirable that the C.AR.ME system should interpret drawing the end of a line near to the starting point, as an attempt to complete the polygon thus providing more support to direct manipulation.

*Figure 6. User task analysis of sub-goal: Polygon drawing.*

(b) The second misconception detected was related to the fact that in order to automatically produce the geometric equivalent shape, the user was requested -through a relevant message- to measure the area of the original polygon (a). Demanding this activity in the frame of an automatic transformation task created confusion to the pupil, as seen in actions 8-10 of the log file (Table 1). Finally the pupil, after measuring the area of (a), produced some equivalent shapes (see Figure 7, tasks 1.2.1, 1.2.5 and table 1 actions 18,19). A comment was added to sub-task 1.2.2.1 in the STM, shown in Figure 7, describing the observed user behavior and the relevant annotation mark ( ! ) was assigned to the task execution.

*Figure 7. Extract of User Task Model. sub-goal: polygon automatic transformation.*

( c) Subsequently, when the pupil attempted to create more equivalent shapes the system prompted her to measure one side of a previously drawn shape, in order to use it as a base. This does not clearly relate to the task objective. The result was that the user could not carry out the task and started to try different ways of interaction ending in a deadlock (actions 20-28,Table 2). Thus, the tasks 1.2.3, 1.2.4, 1.2.6 were marked as unaccomplished in the STM, as shown in Figure 7.

By reviewing the interaction extract we can conclude that due to flaws in system design, only 10 out of 28 listed actions (35%) were related to expected task execution, and finally the given task was not fully accomplished.

User observation analysis both of screenshots and log files produced quite large amount of feedback on usability but the analysis procedure was tedious and time consuming. An automated strategy classification technique using Bayesian belief networks (Tselios et al. 2001) has been proposed in order to classify click stream data, thus facilitating the student goal inferring process.  However more research and experimentation is required in this direction.

Another important finding of this process was that this association of low-level user actions to task and goal models, as these had been expressed in the form of DTM, revealed flaws in the original prototype design and easily associated these flaws to specific tasks and tools. Moreover, the student's task models of various users were compared and the syntactic and semantic deviations of user interaction visualized in reference to the DTM, as shown in Figure 7.

## 6. Comparison of the model-based approach to heuristic evaluation

The proposed design and evaluation framework and the developed tools are compared to the widely used heuristic usability evaluation approach in this section. An independent study was undertaken using the latter approach on the CARME microworld and a comparison of the findings of the two approaches was made.

Heuristic evaluation (HE) focused on design of the user interface in terms of layout consistency, appropriate feedback and minimization of errors rather than evaluation of interaction mechanisms required to carry out specific tasks, related to learning.

In our experiment heuristic evaluation unveiled forty eight (48) usability problems in the CARME microworld. As expected, the majority of usability issues unveiled through heuristic evaluation, concerned interface presentation issues, such as appropriateness of language used in the open file/save dialogs, menu item arrangement, semantics of buttons and icons used, appropriateness of metaphors used and quality of feedback in certain dialogue messages. Application of specific heuristics such as degree of support to user's control and freedom, could not be evaluated accurately and exhaustively, especially in such open problem solving environments which are highly interactive, since we cannot anticipate students' behavior without involving user testing techniques in the evaluation method.

The field-based task modeling approach identified thirty-one (31) usability problems in CARME, from which eighteen (18) where common to those identified through the HE approach.  However, through this approach it was found easier to understand *how* a dialogue across student and environment takes place, during task accomplishment, especially when a novel approach was used. In some cases due to unforeseen strategies, new requirements for tools functionalities emerged, as discussed in section 5.  Examples were the discovery that CARME was lacking of mechanisms to facilitate construction of shapes such as squares and circle drawing, the need of which was identified during the field study.

An essential conclusion of this comparative study was that heuristic evaluation, can miss important flaws and limitations of the software design, since the unveiled problems could hardly be related to the effect on learning tasks. Therefore redesign of educational software based solely on heuristic evaluation may not influence more effective learning task redesign. However the findings of the two approaches seem to have complementary elements. As Karat (1988) proposed,  it is through combination of heuristic evaluation and user testing that one is expected to address both general usability issues and domain-specific problems from actual usage observation.

This was proven in our case as both methods unveiled problems of different nature. In general, heuristic evaluation unveiled problems in the *lexical* level of *expected* interaction relating to the way the environment delivered and received responses during a student interaction rather than on *how* actually this dialogue takes place and missed issues relating with what students *may* need in order to solve specific tasks. Thus heuristic evaluation could play a supportive role to unveil interface design limitations, thus facilitating reasoning and identifying the cause of observed interaction errors.

## 7. Conclusions

The proposed design and evaluation framework which captures the details of student interaction with the problem solving environment both at the cognitive level as well as the keystroke level, supported by the developed tools, is an innovative approach of design and evaluation of open problem solving educational environments. The proposed methodological approach provides flexible and expressive notations, systematic methods to support the specification, analysis and use of task models, support for the reuse of good design solutions and tools to support the modeling framework.

As demonstrated in the described case study, detailed analysis of students' problem solving behavior, building of the individual student task models and aggregating these in a single revised designer task model, can lead to a bottom up re-design phase following the original top-down design. The observations made during this phase proved to be very rich, and their capturing and classification under the syntactic and semantic re-design perspectives, following this approach, define a solid framework for iterative design of open learning environments.

The proposed model-based techniques permit deeper understanding of the nature of tasks through their explicit representation and annotation during their decomposition and systematic consistency control across task structures. Additional advantages of this framework, were introduced by the CMTool used to carry out the modelling process. These are the possibility of deriving quantitative and qualitative results concerning the problem solving process at both at the individual student level and at the group level. Using Visual Query Construction environment, extensive analysis on intra-individual student differences, in terms of strategy acquisition, usage of strategies and difficulties found is supported.

Some shortcomings of the proposed approach relate to the difficulty of inferring the cognitive model of the user from the keystroke behavior. This problem was tackled by using adequate field evaluation protocols that involve deep interaction of evaluators with the user during the field study (e.g. think-aloud protocol, interviews etc).

Finally, this approach can be integrated with a constructivist perspective of learning evaluation process as it supports the study of the development of the individual pupil strategies. From an educational perspective this approach supports the study of each individual pupil strategies in relation to the general student model constructed by taking into account all pupils' strategies that participated in a particular study. Open problem-solving educational environments in general suffer from the lack of adequate learning evaluation mechanisms, since identification of correct solutions is often a tedious process. By modeling the user interaction and judge it against the original model's possible solutions, as provided by educational experts, facilitates validation of the solutions provided by the problem solvers.

## Acknowledgements

The reported research has been funded by the GSRT the research project PENED 99ED234 “Intelligent Distance learning Tutoring Systems” and the European Union project IST2000-25385 Modelingspace. Special thanks are due to two anonymous reviewers for comments on earlier draft.

## References

Abowd, G.D. (1992) Using formal methods for the specification of user interfaces. Proceedings of the Second Irvine Software Symposium, ISS 92:109-130.

Ainsworth, L. & Pendlebury, G., (1995) Task-based contributions to the design and assessment of the man-machine interfaces for a pressurized water reactor, Ergonomics 38 (3) : 462-474.

Avouris, N.M. Tselios, N. & Tatakis E.C. (2001) Development and Evaluation of a Computer-based Laboratory teaching tool, J. Computer Applications in Engineering Education, vol. 9 (1), pp. 8-19.

Balacheff, N. & Kaput, J. (1996). Computer-based learning environments in mathematics. In A. J. Bishop, K. Klements, C. Keitel, J. Kilpatric and C. Laborde (Eds), International Handbook on Mathematics education (pp. 469-501). Dortdrecht: Kluwer.

Barnard, P.J. (1987) Cognitive resources and the learning of human-computer dialogues, In Carroll, J. M. (eds)  Interfacing thought : Cognitive aspects of human computer interaction; MIT Press.

Card, S. Moran, T. & Newell, A. (1983) The Psychology of Human Computer Interaction. Lawrence Erlbaum Associates.

Diaper, D. (1989) Task analysis and systems analysis for software development. Interacting with computers 4(1), pp.124-139.

Gong, R. & Kieras, D. (1994) A Validation of the GOMS Model Methodology in the Development of a Specialized, Commercial Software Application, Proceedings ACM CHI 1994 (Boston Massachusetts, April 24-28), pp. 351-357.

Haunold, P. & Kuhn, W. (1994) A Keystroke Level of a Graphic Application : Manual Map Digitizing, Proceedings ACM CHI 1994, pp. 337-343.

Inkpen, K. (1997) Three Important Research Agendas for Educational Multimedia: Learning, Children, and Gender. AACE World Conference on Educational Multimedia and Hypermedia 97, Calgary, AB, June 1997, pp 521-526.

John, B. & Kieras, D. (1996) Using GOMS for user interface design and evaluation: Which technique? , Proc. ACM Transactions on Computer-Human Interaction vol. 3(4), pp. 287-319.

John, B. & Kieras, D. (1996a) The GOMS family of user interface analysis techniques: Comparison and contrast, ACM Transactions on Computer-Human Interaction, vol. 3(4), pp. 320-351.

John, B. & Vera, A. (1992) A GOMS Analysis of a Graphic Machine Paced, highly Interactive task. Proceedings ACM CHI 92, pp. 251-258.

John, B. & Wayne, G. (1994) GOMS Analysis for Parallel Activities. Tutorial, Proc. ACM Transactions on Computer-Human Interaction 1994, pp. 395-396.

Johnson, H. & Johnson, P. (1991) Task knowledge structures: Psychological basis and integration into system design. Acta Psychologica 78, North Holland, pp. 3-26.

Kaput, J.J. (1987) Representation systems and mathematics. In Problems of representation in teaching and learning of mathematics ,C. Janvier (Eds) London: Lawrence Erlbaum associates, pp.19-26.

Karat, J. (1988) Software Evaluation Methodologies, Handbook of Human-Computer Interaction, M. Helander (ed.), Elsevier Science Publishers, B.V. (North-Holland), pp. 891-903.

Kieras, D. (1996) Task analysis and the design of functionality. In CRC Handbook of Computer Science and Engineering, CRC Press.

Kordaki, M. & Potari, D. (1998). A learning environment for the conservation of area and its measurement: a computer microworld, Computers and Education, 31, pp. 405-422.

Kordaki, M. & Avouris, N (2001) Modeling in Design and Evaluation of Open Learning Environments, Computers and Education (submitted).

Laborde, J.M. & Strasser, R. (1990) Cabri-Geometry: A microworld of geometry for guided discovery learning. ZDM, 5:171-177.

Lansdale, M.W. & Ormerod, T.C. (1994) , Understanding Interfaces: A Handbook of Human-Computer Interaction, Academic Press, London.

Lim, K.Y (1996) Structured task analysis: an instantiation of the MUSE method for usability engineering, Interacting with Computers, vol. 8(1), pp. 31-50.

Mayes J. T.,Fowler C. J.,(1999) Learning Technology and Usability: A Framework for Understanding Courseware Usability and Educational Software Design.Interacting with Computers 1999 v.11 n.5 p.485-497

Moran, T. P. (1981) The command language grammar: a representation for the user interface of interactive computer system, Int. J. of Man-Machine Studies 15, pp. 3-50.

Nielsen J. (1993) Usability Engineering, Academic Press, London.

Olson, J. & Olson, G. (1990) The Growth of Cognitive Modeling in Human-Computer Interaction Since GOMS, Human Computer Interaction, Vol.5, pp. 221-265.

Orhun, E. (1995) Design of Computer-Based Cognitive Tools. In: diSessa, A. A.; Hoyles, C. & Noss, R. (Eds.) Computers and Exploratory Learning. Berlin: Springer, pp. 305-320.

Papert, S. (1980) Mindstorms: Pupils, Computers, and Powerful Ideas. New York: Basic Books.

Paterno', F. & Ballardin, G. (2000) RemUSINE: a Bridge between Empirical and Model-based Evaluation when Evaluators and Users are Distant, Interacting with Computers, Vol.13, N.2, pp. 151-167.

Patterno' F. & Mancini, C. (2000) Model-based design of Interctive Applications, ACM Intelligence, winter 2000, pp. 27-37.

Richardson, J. Ormerod, T. & Shepherd A. (1998) The role of task analysis in capturing requirements for interface design , Interacting with computers vol. 9(2), pp. 367-384.

Sedig, K., Klawe, M., & Westrom, M. (2001) Role of Interface Manipulation Style and Scaffolding on Cognition and Concept Learning in Learnware. ACM Transactions on Computer-Human Interaction, 8 ( 1), March 2001, pp 34–59.

Shepherd, A. (1989); Analysis and training in information technology task. In Diaper, D.(ed.); Task Analysis for human computer interaction, Ellis Horwood Limited, pp.15-55.

Tselios, N. Maragoudakis, M. Avouris, N. Fakotakis, N. Kordaki, M. (2001a) Automatic diagnosis of student problem solving strategies using Bayesian Networks, 5th Panhellenic Conf. on Didactics of Mathematics and Informatics in Education, Thessaloniki, 12-14 October 2001.

Tselios N., Avouris N., Dimitracopoulou A, (2001b), Evaluation of Distance-learning Environments: Impact of Usability on Student Performance, Int. Journal of Educational Telecommunications, forthcoming.

Umbers, I.G., Reirsen, C.S. (1995) Task analysis in support of the design and development of a nuclear power plant safety system, Ergonomics 38 (3) 443-454.

van Welie M.,van der Veer G.C., Eliens A.(1998) Euterpe - Tool support for analyzing cooperative environments, Proceedings of the 9th European Conference on Cognitive Ergonomics , August 24-26, 1998, Limerick, Ireland.

Von Glasersfeld, E. (1987) Learning as a constructive activity. In Janvier, C. (Ed.) Problems of representation in teaching and learning of mathematics London: Lawrence Erlbaum, pp. 3-18.

Walsh, P. A. (1989); Analysis for Task Object Modeling (ATOM) towards a method of integrating task analysis with Jackson System Development for user interface software design. In Diaper, D.(eds); Task Analysis for human computer interaction, Ellis Horwood, pp.186-209.

Wilson, S. , Johnson, P., Kelly, C. , Cunningham, J., Markopoulos, P. (1993) Beyond Hacking: a model based approach to user interface design, in Proc. of HCI'93**,** J. Alty, D. Diaper and S. Guest (eds) Cambridge University press, pp 217-231.

***Abbreviations used***

| | |
|---|---|
| CMTool : | Cognitive Modeling Tool |
| DTM : | Designer Task Model |
| GOMS: | Goals Operators Methods Selection Rules |
| HCI : | Human Computer Interaction |
| HTA: | Hierarchical Task Analysis |
| STM : | Student Task Model |
| TA : | Task Analysis |
| UATool : | Usability Analyzer Tool |

## APPENDIX

### *CMTool Plans annotation.*

Temporal relationships of subtasks are described through *plans*, shown next to a task node. The plans contain the following operators:

AND and OR are used to describe tasks that are executed together or by selecting one instead of another, respectively. Instead of AND the comma mark (,) can be used to describe tasks that could be executed in any order.

IF conditional task execution

SEQ or THEN or FOLLOWED BY are used to describe tasks that must be executed in sequence

PAR is used to indicate tasks that can be carried out in parallel.

*N repeat task N times

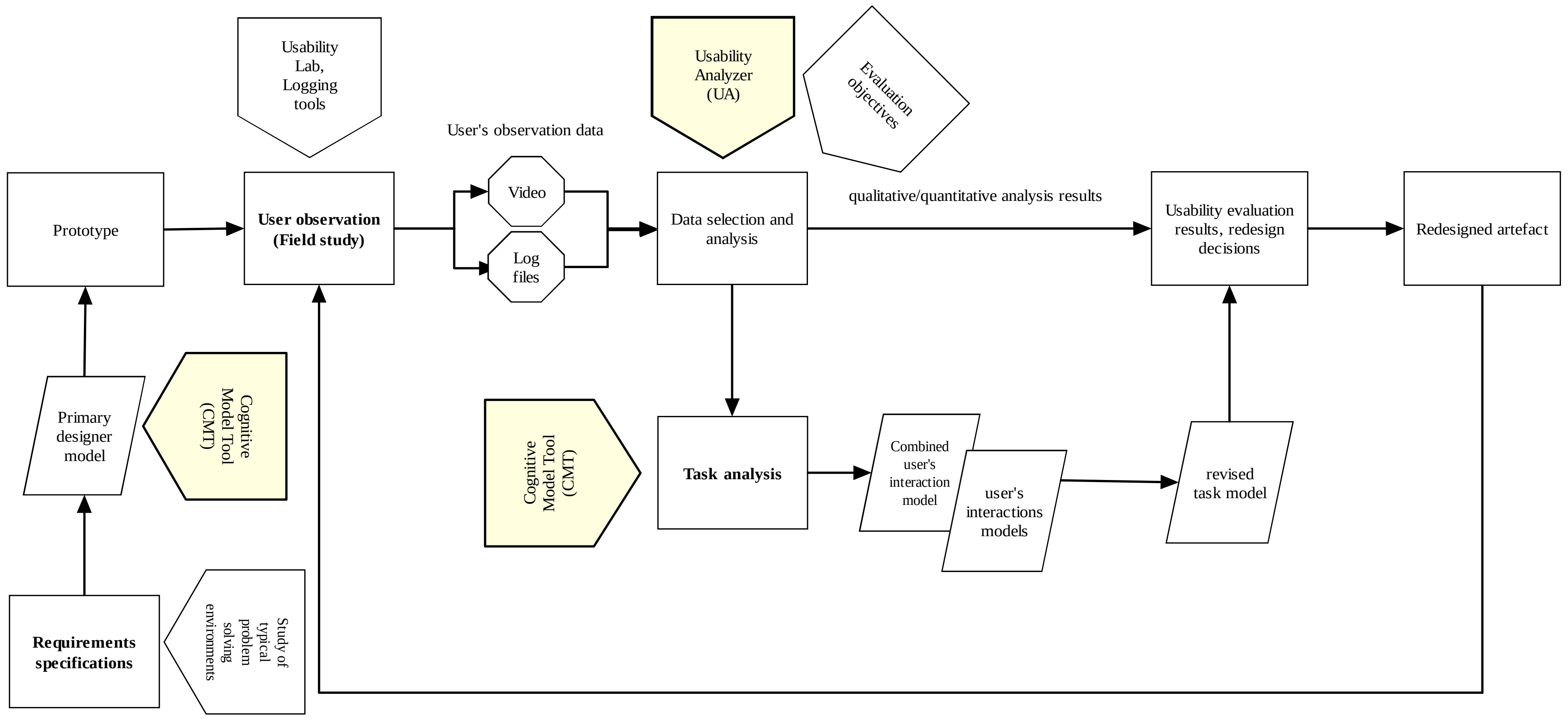
### *Tokens for annotation of deviations in task models*

(!) marks a non-destructive syntactic deviation,

(x!) marks a not-completed task execution.

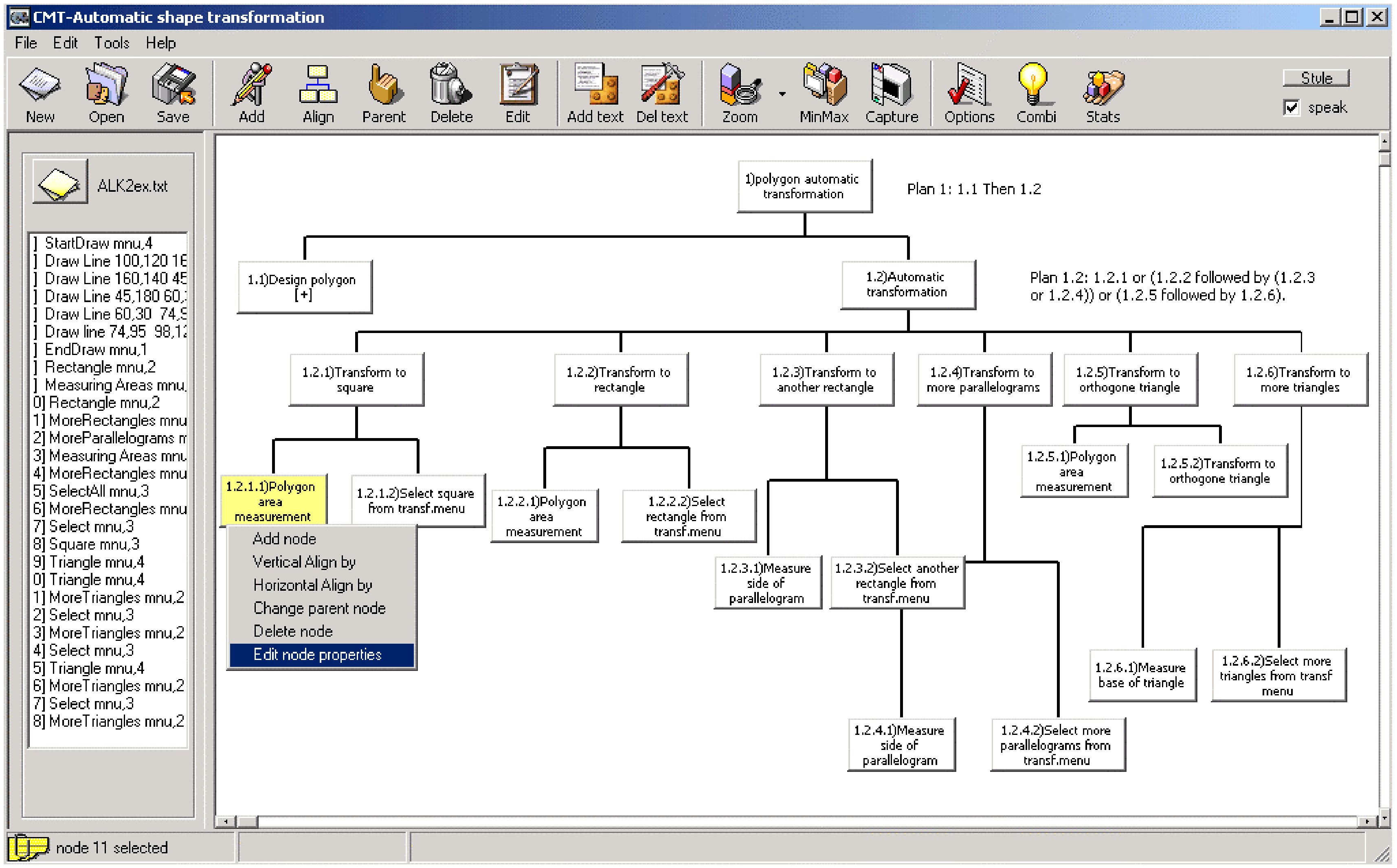
{task-id} Introduction of new unforeseen tasks in a plan by a user

<task-id> Tasks that should be supported in a different way in order to meet the expectations of the user



Usability Lab, Logging tools
User's observation data
Usability Analyzer (UA)
Evaluation objectives
Prototype
User observation (Field study)
Video
Log files
Data selection and analysis
qualitative/quantitative analysis results
Usability evaluation results, redesign decisions
Redesigned artefact
Primary designer model
Cognitive Model Tool (CMT)
Cognitive Model Tool (CMT)
Task analysis
Combined user's interaction model
user's interactions models
revised task model
Requirements specifications
Study of typical problem solving environments

*Figure 1 Outline of the evaluation framework.*

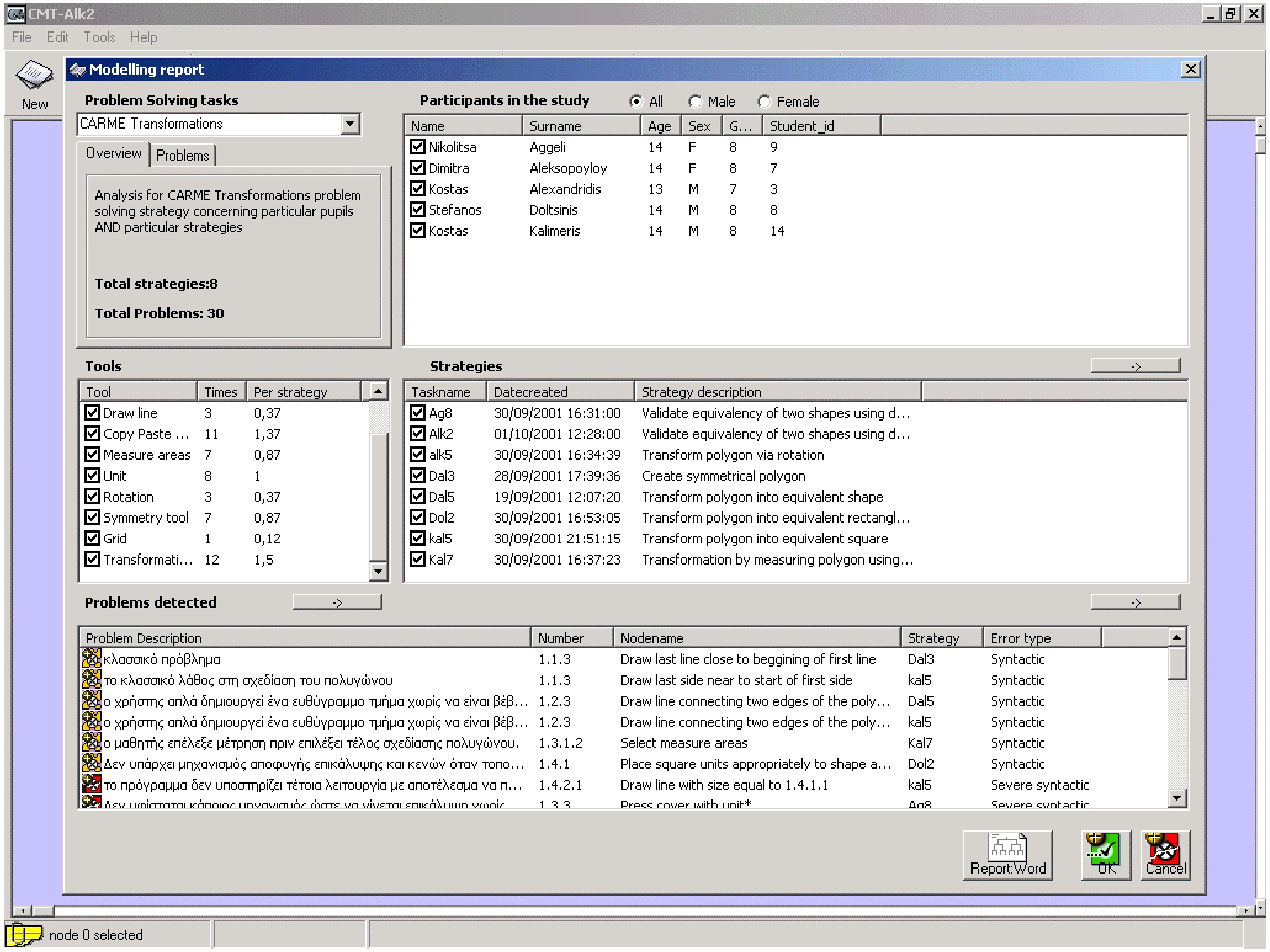

CMT-Automatic shape transformation
File Edit Tools Help
New Open Save Add Align Parent Delete Edit Add text Del text Zoom MinMax Capture Options Combi Stats
Style
speak
ALK2ex.txt
1)polygon automatic transformation
1.1)Design polygon [+]
1.2)Automatic transformation
1.2.1)Transform to square
1.2.2)Transform to rectangle
1.2.3)Transform to another rectangle
1.2.4)Transform to more parallelograms
1.2.5)Transform to orthogone triangle
1.2.6)Transform to more triangles
1.2.1.1)Polygon area measurement
1.2.1.2)Select square from transf.menu
1.2.2.1)Polygon area measurement
1.2.2.2)Select rectangle from transf.menu
1.2.5.1)Polygon area measurement
1.2.5.2)Transform to orthogone triangle
1.2.3.1)Measure side of parallelogram
1.2.3.2)Select another rectangle from transf.menu
1.2.6.1)Measure base of triangle
1.2.6.2)Select more triangles from transf menu
1.2.4.1)Measure side of parallelogram
1.2.4.2)Select more parallelograms from transf.menu
Add node
Vertical Align by
Horizontal Align by
Change parent node
Delete node
Edit node properties
node 11 selected

Plan 1: 1.1 Then 1.2

Plan 1.2: 1.2.1 or (1.2.2 followed by (1.2.3 or 1.2.4)) or (1.2.5 followed by 1.2.6).

```
] StartDraw mnu,4
] Draw Line 100,120 16
] Draw Line 160,140 45
] Draw Line 45,180 60,
] Draw Line 60,30  74,9
] Draw line 74,95  98,1
] EndDraw mnu,1
] Rectangle mnu,2
] Measuring Areas mnu,
0] Rectangle mnu,2
1] MoreRectangles mnu
2] MoreParallelograms m
3] Measuring Areas mnu
4] MoreRectangles mnu
5] SelectAll mnu,3
6] MoreRectangles mnu
7] Select mnu,3
8] Square mnu,3
9] Triangle mnu,4
0] Triangle mnu,4
1] MoreTriangles mnu,2
2] Select mnu,3
3] MoreTriangles mnu,2
4] Select mnu,3
5] Triangle mnu,4
6] MoreTriangles mnu,2
7] Select mnu,3
8] MoreTriangles mnu,2
```

*Figure 2. CMTool : The main task model building space. Part of the designer task model(DTM) of the task discussed in section 5 is shown*



**Problem Solving tasks**

CARME Transformations

Overview | Problems

Analysis for CARME Transformations problem solving strategy concerning particular pupils AND particular strategies

**Total strategies:8**

**Total Problems: 30**

**Participants in the study** [✓ All] Male Female

| Name | Surname | Age | Sex | G... | Student_id |
|---|---|---|---|---|---|
| ✓ Nikolitsa | Aggeli | 14 | F | 8 | 9 |
| ✓ Dimitra | Aleksopoyloy | 14 | F | 8 | 7 |
| ✓ Kostas | Alexandridis | 13 | M | 7 | 3 |
| ✓ Stefanos | Doltsinis | 14 | M | 8 | 8 |
| ✓ Kostas | Kalimeris | 14 | M | 8 | 14 |

**Tools**

| Tool | Times | Per strategy |
|---|---|---|
| ✓ Draw line | 3 | 0,37 |
| ✓ Copy Paste ... | 11 | 1,37 |
| ✓ Measure areas | 7 | 0,87 |
| ✓ Unit | 8 | 1 |
| ✓ Rotation | 3 | 0,37 |
| ✓ Symmetry tool | 7 | 0,87 |
| ✓ Grid | 1 | 0,12 |
| ✓ Transformati... | 12 | 1,5 |

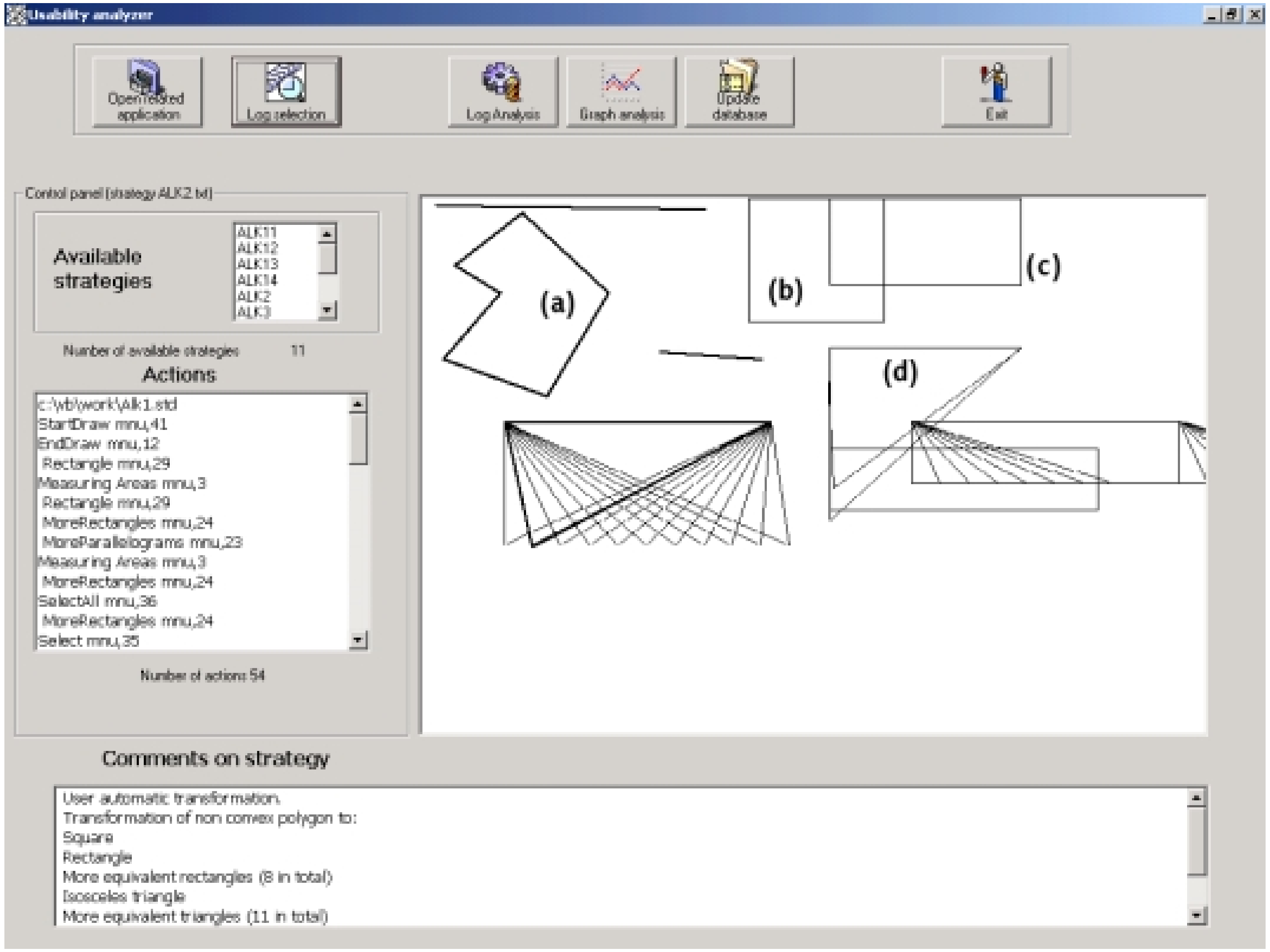
**Strategies**

| Taskname | Datecreated | Strategy description |
|---|---|---|
| ✓ Ag8 | 30/09/2001 16:31:00 | Validate equivalency of two shapes using d... |
| ✓ Alk2 | 01/10/2001 12:28:00 | Validate equivalency of two shapes using d... |
| ✓ alk5 | 30/09/2001 16:34:39 | Transform polygon via rotation |
| ✓ Dal3 | 28/09/2001 17:39:36 | Create symmetrical polygon |
| ✓ Dal5 | 19/09/2001 12:07:20 | Transform polygon into equivalent shape |
| ✓ Dol2 | 30/09/2001 16:53:05 | Transform polygon into equivalent rectangl... |
| ✓ kal5 | 30/09/2001 21:51:15 | Transform polygon into equivalent square |
| ✓ Kal7 | 30/09/2001 16:37:23 | Transformation by measuring polygon using... |

**Problems detected**

| Problem Description | Number | Nodename | Strategy | Error type |
|---|---|---|---|---|
| κλασσικό πρόβλημα | 1.1.3 | Draw last line close to beggining of first line | Dal3 | Syntactic |
| το κλασσικό λάθος στη σχεδίαση του πολυγώνου | 1.1.3 | Draw last side near to start of first side | kal5 | Syntactic |
| ο χρήστης απλά δημιουργεί ένα ευθύγραμμο τμήμα χωρίς να είναι βέβ... | 1.2.3 | Draw line connecting two edges of the poly... | Dal5 | Syntactic |
| ο χρήστης απλά δημιουργεί ένα ευθύγραμμο τμήμα χωρίς να είναι βέβ... | 1.2.3 | Draw line connecting two edges of the poly... | kal5 | Syntactic |
| ο μαθητής επέλεξε μέτρηση πριν επιλέξει τέλος σχεδίασης πολυγώνου. | 1.3.1.2 | Select measure areas | Kal7 | Syntactic |
| Δεν υπάρχει μηχανισμός αποφυγής επικάλυψης και κενών όταν τοπο... | 1.4.1 | Place square units appropriately to shape a... | Dol2 | Syntactic |
| το πρόγραμμα δεν υποστηρίζει τέτοια λειτουργία με αποτέλεσμα να π... | 1.4.2.1 | Draw line with size equal to 1.4.1.1 | kal5 | Severe syntactic |
| Δεν υφίσταται κάποιος μηχανισμός ώστε να γίνεται επικάλυψη χωρίς | 1.3.3 | Press cover with unit* | Ag8 | Severe syntactic |

Figure 3. Visual Query Construction environment.



Number of available strategies 11

```
c:\wb\work\Alk1.std
StartDraw mnu,41
EndDraw mnu,12
 Rectangle mnu,29
Measuring Areas mnu,3
 Rectangle mnu,29
 MoreRectangles mnu,24
 MoreParallelograms mnu,23
Measuring Areas mnu,3
 MoreRectangles mnu,24
SelectAll mnu,36
 MoreRectangles mnu,24
Select mnu,35
```

Number of actions: 54

User automatic transformation.
Transformation of non convex polygon to:
Square
Rectangle
More equivalent rectangles (8 in total)
Isosceles triangle
More equivalent triangles (11 in total)

*Figure 4.Usability Analyzer : On the left the log file of the user is shown. In the main window the corresponding user screenshots can be studied, at the bottom evaluator's comments are included. The data shown are related to the task discussed in the case study of section 4.*

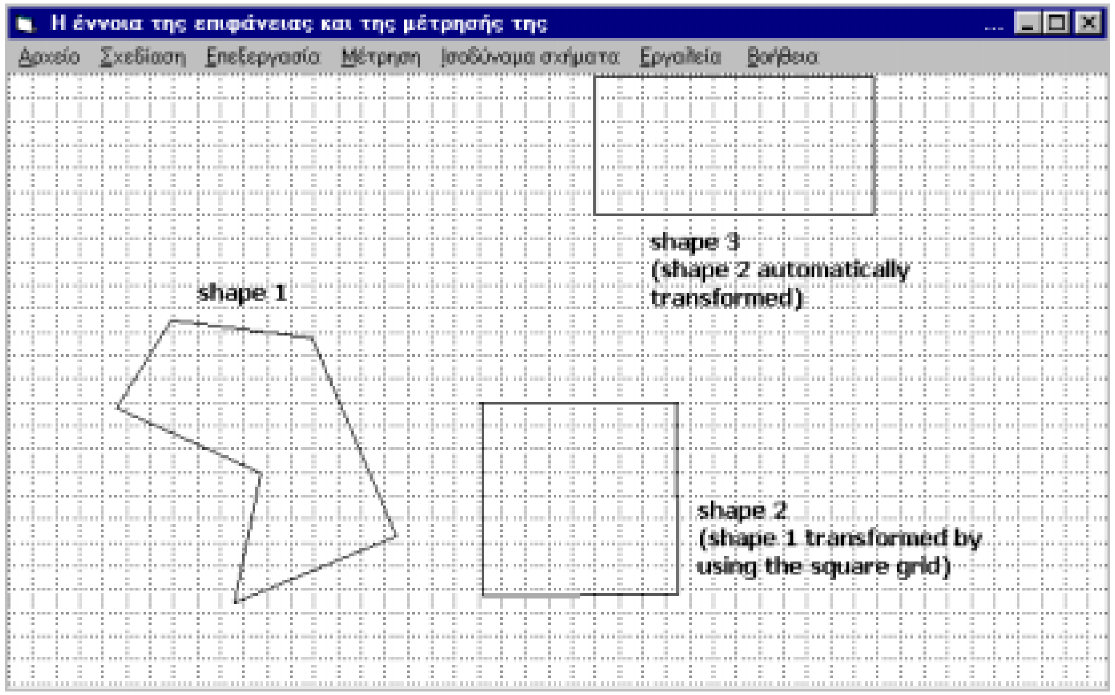


*Figure 5. Screen of the CARME environment : using the concepts of conservation and of area measurement in integration*

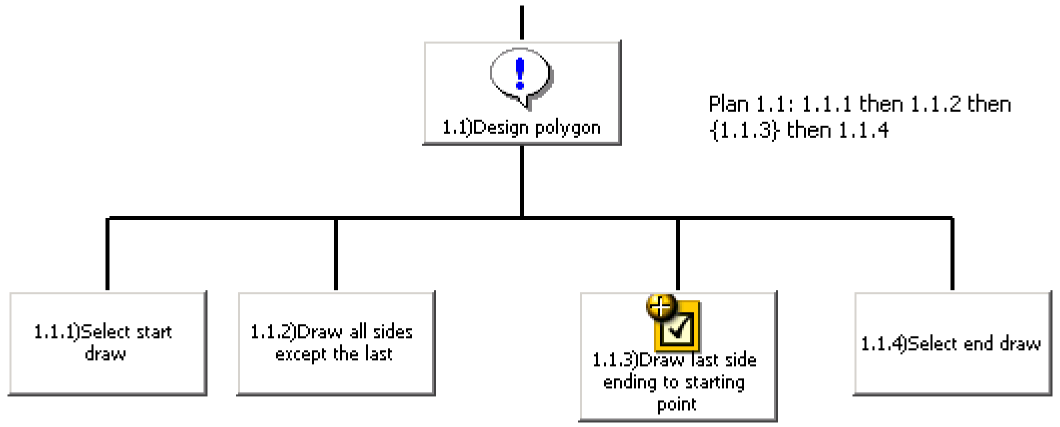
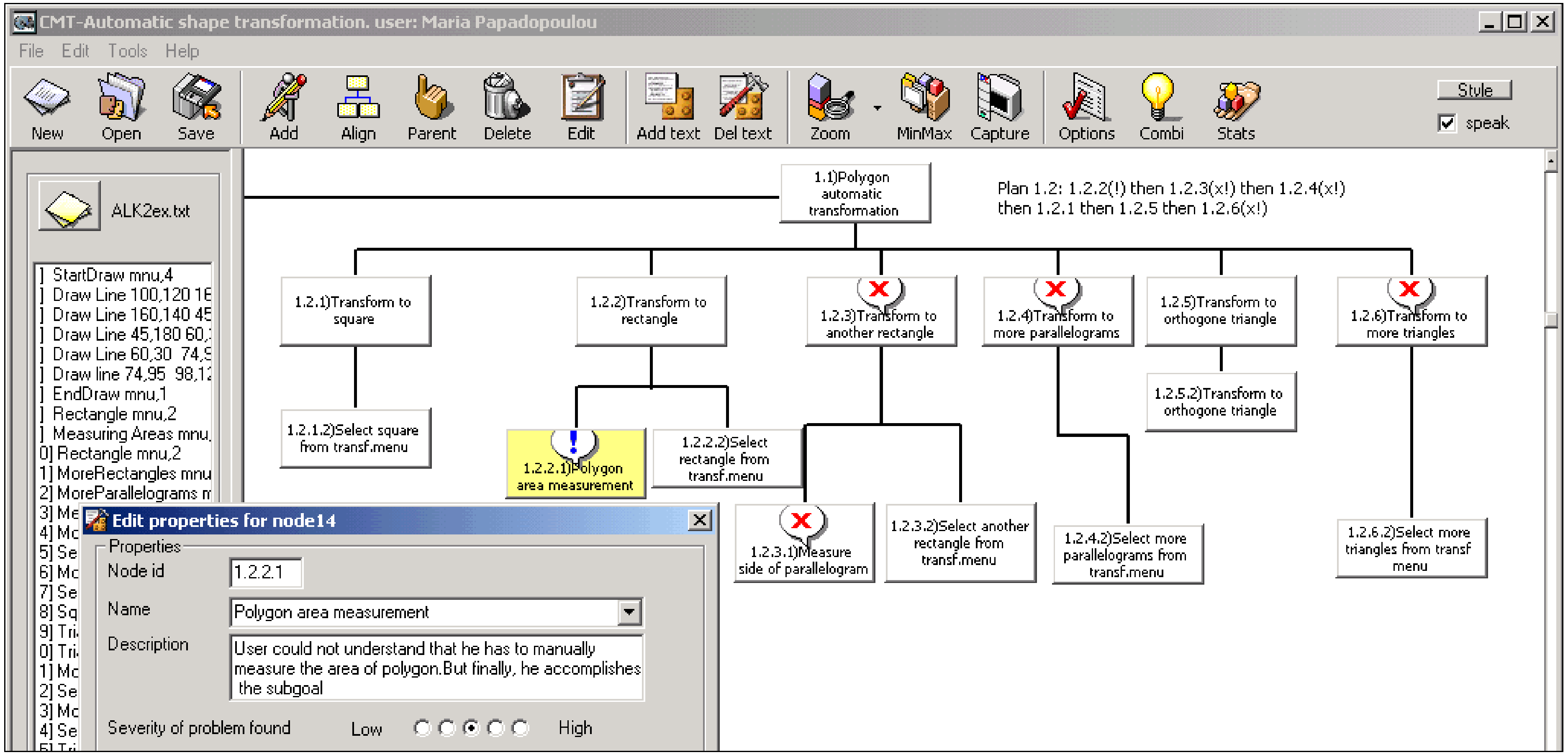

1.1)Design polygon
Plan 1.1: 1.1.1 then 1.1.2 then {1.1.3} then 1.1.4
1.1.1)Select start draw
1.1.2)Draw all sides except the last
1.1.3)Draw last side ending to starting point
1.1.4)Select end draw

*Figure 6. User task analysis of sub-goal: Polygon drawing.*


CMT-Automatic shape transformation. user: Maria Papadopoulou
File
Edit
Tools
Help
New
Open
Save
Add
Align
Parent
Delete
Edit
Add text
Del text
Zoom
MinMax
Capture
Options
Combi
Stats
Style
speak
ALK2ex.txt
1.1)Polygon automatic transformation
1.2.1)Transform to square
1.2.2)Transform to rectangle
1.2.3)Transform to another rectangle
1.2.4)Transform to more parallelograms
1.2.5)Transform to orthogone triangle
1.2.6)Transform to more triangles
1.2.1.2)Select square from transf.menu
1.2.2.1)Polygon area measurement
1.2.2.2)Select rectangle from transf.menu
1.2.3.1)Measure side of parallelogram
1.2.3.2)Select another rectangle from transf.menu
1.2.4.2)Select more parallelograms from transf.menu
1.2.5.2)Transform to orthogone triangle
1.2.6.2)Select more triangles from transf menu
Edit properties for node14

Plan 1.2: 1.2.2(!) then 1.2.3(x!) then 1.2.4(x!) then 1.2.1 then 1.2.5 then 1.2.6(x!)

```
] StartDraw mnu,4
] Draw Line 100,120 16
] Draw Line 160,140 45
] Draw Line 45,180 60,
] Draw Line 60,30  74,9
] Draw line 74,95  98,12
] EndDraw mnu,1
] Rectangle mnu,2
] Measuring Areas mnu,
0] Rectangle mnu,2
1] MoreRectangles mnu
2] MoreParallelograms m
3] Me
4] Mc
5] Se
6] Mc
7] Se
8] Sq
9] Tri
0] Tri
1] Mc
2] Se
3] Mc
4] Se
```

Properties

| | |
|---|---|
| Node id | 1.2.2.1 |
| Name | Polygon area measurement |
| Description | User could not understand that he has to manually measure the area of polygon.But finally, he accomplishes the subgoal |
| Severity of problem found | Low ○ ○ ◉ ○ ○ High |

*Figure 7. Extract of User Task Model. sub-goal: polygon automatic transformation.*

| | Tool name | # used | Per strategy |
|---|---|---|---|
| 1 | Cognitive tasks | 61 | 7,62 |
| 2 | Polygon | 30 | 3,75 |
| 3 | Transformations | 12 | 1,5 |
| 4 | Copy/Paste Tools | 11 | 1,37 |
| 5 | Units | 8 | 1 |
| 6 | Measure areas | 7 | 0,87 |
| 7 | Symmetry tool | 7 | 0,87 |
| 8 | Draw line | 3 | 0,37 |
| 9 | Rotation | 3 | 0,37 |
| 10 | Grid | 1 | 0,12 |
| | **Total** | **143** | **17,875** |

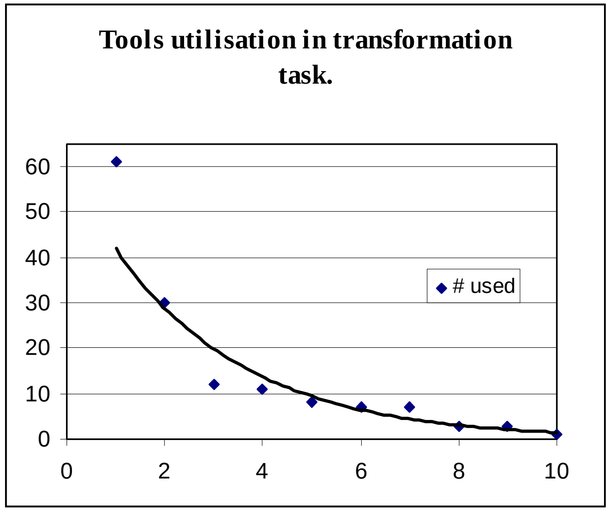


*Table 1. Frequency of tool usage in transformation task.*

| Example Log file | Comments |
|---|---|
| 1]  StartDraw mnu,41 | |
| 2]  Draw Line 100,120 160,140 | |
| 3]  Draw Line 160,140 45,180 | *Actions [1-7]* : design of the reference polygon (a) of figure 1. Last segment drawn through action [6]. |
| 4]  Draw Line 45,180 60,30 | |
| 5]  Draw Line 60,30  74,95 | |
| 6]  Draw line 74,95  98,120 | *Actions [8-10]* : Selection of menu option automatic transformation of rectangle. The system responded with request to measure area of (a) first. The user selected area measurement [9] and then proceeded with Rectangle creation [10]. Successful completion of task (1.2.2). |
| 7]  EndDraw mnu,12 | |
| 8]  Rectangle mnu,29 | |
| 9]  Measuring Areas mnu,3 | |
| 10] Rectangle mnu,29 | |
| 11] MoreRectangles mnu,24 | |
| 12] MoreParallelograms mnu,23 | *Actions [11-17]* : The user attempts unsuccessfully  to build more equivalent shapes, i.e.  tasks (1.2.3) and (1.2.4). In order to pursue these goals, the user needs to achieve goals (1.2.3.1) and (1.2.4.1) first. This is not clear by system response. |
| 13] Measuring Areas mnu,3 | |
| 14] MoreRectangles mnu,24 | |
| 15] SelectAll mnu,36 | |
| 16] MoreRectangles mnu,24 | |
| 17] Select mnu,35 | |
| 18] Square mnu,39 | *Action [18]* : The user completes successfully task (1.2.1) |
| 19] Triangle mnu,44 | |
| 20] Triangle mnu,44 | |
| 21] MoreTriangles mnu,25 | *Action [19*] : The user completes successfully task (1.2.5) |
| 22] Select mnu,35 | |
| 23] MoreTriangles mnu,25 | |
| 24] Select mnu,35 | *Actions [20-28]* : Unsuccessful attempt to achieve task (1.2.6). The system requests first manual measurement of the base of the triangle. The user abandons the goal. |
| 25] Triangle mnu,44 | |
| 26] MoreTriangles mnu,25 | |
| 27] Select mnu,35 | |
| 28] MoreTriangles mnu,25 | |

Table 2. Example log file and comments on the interaction behavior.