STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LI**, Number 2, 2006

OPERATIONAL SEMANTICS OF TASK MODELS

ADRIANA TARŢA AND SIMONA MOTOGNA

Abstract. This paper proposes an operational semantic approach for the task models. Task models are used in the process of user centered software design to represent the work structure and the sequence of steps needed to reach a goal. The goal of our approach is to develop inference rules which will reflect the changes on the presentation aspects of an application when a certain temporal operator occurs. By using the proposed inferece rule we will show that when the deductive process is ended the enabled task sets are determined.

1. Introduction

Today, computers are used in almost every domain of our life. The success or failure of a software system is given by the support the system gives to its user in performing their tasks efficiently and with satisfaction. These features of a software systems are components of a software quality measure, called *usability*. In order to obtain a high usability level, a user centered approach in the design of interactive system should be used. In this paper we will discuss a method used in the design of interactive system called task-based design. The task-based design relies on the analysis of the tasks the users perform. The result of the task analysis is represented by task models. In this paper we will present a formal approach of task models based on operational semantics [10]. The paper is structured as follows: Section 2 presents the basic elements of task-based design, Section 3 presents some introductory notions about operational semantics followed by the presentation of our approach in describing task models using operational semantics, Section 4 presents few examples of applying operational semantics on some task trees and Section 5 presents the conclusions of our research and further work on our approach.

Received by the editors: November 28, 2006.

2000 *Mathematics Subject Classification.* 68-xx, 68Qxx.

1998 *CR Categories and Descriptors.* [**F.4.1**]: Theory of Computation – *Mathematical Logic*; [**H.5.2** [Information Systems]: Information Interfaces and Presentation – *User Interfaces*.

2. Task-based design

The first step in task based design of interactive systems is *task analysis*. Task analysis is the process of gathering data about tasks people perform and acquiring a deep understanding of it. The process of structuring data and gaining insight into the data is called *task modeling* [14]. Work structure is one of the most important aspects in task analysis. The design of an interactive system usually means restructuring the work and removing or adding tasks. Work structure can be captured in a task decomposition tree [15]. The tree forms a hierarchy where the high level tasks are found at the top of the tree and the most basic tasks are at the leaf nodes. In the representations proposed over time ([14], [7]), the tree are enhanced *constructors* that indicate time relationship between tasks. In the work structure model, the root of the tree is a goal with possibly some subgoals. Connected to goals are tasks and tasks can be connected on the same level by temporal operators. When designing for usability, the work structure is important for developing the most appropriate interaction structure and functionality.

Many task analysis methods have been developed and used in the design of software systems: HTA [1], GOMS [3], TAG [9], MAD [11], most of them using a textual formal notation. For complex systems, these methods were impossible to be used. In 1996, a new approach in task based design has been developed, consisting in a method for analysing the groupware work globally, not individually. This new method has been called GTA (Groupware Task Analysis) [13] and based on GTA a design method called DUTCH (Designing for Users and Tasks from Concepts to Handles) [16] has been developed. The task models (trees) developed using GTA used *constructors* to specify the time relationships between tasks. For the average usage, the following time relationships have proved sufficient: Concurrent, Choice, AnyOrder, Succesive and * (iteration) [14]. These constructors were: SUCC for sequential tasks, PAR for concurrent tasks, CHOICE for alternate task, * for iterative tasks and ANY for independent order tasks [14]. CTT (ConcurTaskTrees), another approach in task modeling, uses a more formal time relationship specification using LOTOS (Language Of Temporal Ordering Specification) operators [5]. The operators taken from LOTOS are: enabling, disabling, parallel composition (interleaving and full synchronization), choice, order independence and iteration. In the following, we will present the definition of each of the operators mentioned above. The original definitions are expressed in terms of processes [2]. For the subject of our paper, we will use the term *task* instead of using the term *process*.

The *choice operator* - denoted by [] - is defined as follows: if T1 and T2 are two tasks, then T1[]T2 denotes a task which behaves like T1 or T2.

The *interleaving operator* - denoted by ||| - if T1 and T2 are two tasks ready for an action (t1 and t2), then both action orderings (t1 after t2 ot t2 after t1) are possible.

The *synchronization operator* - denoted by || - T1 and T2 have to synchronize on some actions in order to exchange information.

The *sequential composition* (enabling) operator - denoted by - $\gg$ - T1 $\gg$ T2 - if T1 terminates successfully, then the execution of T2 is enabled.

The *disabling operator* - denoted by [> - T1 [>T2 - if T2 is started, then T1 is never performed.

The *iteration operator* - denoted by * - $T1^*$ - means that task T1 is iterative.

In the following subsection we will use operational semantics in order to systematically derive the actions (subtasks) that a task may perform from the structure of expression itself. The goal of this approach is to determine the Enabled Task Sets (ETS) [4], which will lead us to a *presentation model* of the user interface of an interactive systems.

3. Operational semantics for temporal operators

Operational semantics has been successfully used in specifying different programming languages. The principles can be applied with a few modifications for our goals.

We will denote the set of temporal operators by $\mathcal{O} = \{[], >>, [>, ||, |||, | = |, *\}$. The prority order among operators is: choice operator > parallel composition operators (interleaving, synchronization) > disabling operator > order independence operator> enabling operator [8].

Our aim is that starting from a task tree to obtain in a rigourous way the *enabled tasks sets* (tasks which are enabled at the same time) that will correspond to the user interface of an interactive system. Having the enabled tasks sets, we will able to determine the widgets of the user interface based on task types. Tasks types can be: editing, monitoring, selection of a single choice or selection of multiple choices and control [14]. In the following we will describe our approach in identifying the enabled tasks sets based on the operational semantics.

We will give the definition of a task tree starting from the definition of the tree concept [6]. A *task tree* has a root node (representing the goal of the task performance). Each node is either a leaf (a unit task) or an internal node (representing a subtask). An internal node has one or more children nodes and is called the parent of its child nodes. All children of the same node are siblings. Every two siblings are related by a temporal operator belonging to the set $\mathcal{O}$.

We will use Haskell-like data definitions to define abstract syntax formally:

```
Op=[](ST,ST) | ||(ST,ST)| [> (ST,ST)| >> (ST,ST)| ||| (ST,ST)| *(ST)
```

A task sequence may be defined as

$ST \rightarrow T\ op\ T \mid T\ op\ ST \mid T$

$T \rightarrow task$

$op \rightarrow \gg \mid [] \mid \parallel \mid ||| \mid [> \mid *$.

The *state* of the abstract machine for task trees has a *stack of tasks* and an *enabled task sets (ETS) collection* denoted by $\mathcal{E}$ where the enabled tasks sets generated by the execution of the abstract machine are saved.

The structured operators semantics for task models defines a relation "$\rightarrowtriangle$" which means "is transformed by a single execution step into" [12]. We define this relation by means of inference rules. An expression like "$[\![t_1]\!]\mathcal{E} \rightarrowtriangle [\![skip]\!]\mathcal{E}$" can be read as "execution of task t_1 with the enabled task sets $\mathcal{E}$ means execution of *skip* (no action) with the enabled task sets $\mathcal{E}$". An expression like "$[\![t_1]\!]\mathcal{E} \rightarrowtriangle [\![t_1']\!]\mathcal{E}'$" can be read as "execution of task t_1 with the enabled task sets $\mathcal{E}$ means execution of subtask t_1' with the enabled task sets $\mathcal{E}'$".

3.1. **Enable operator semantics.** When two tasks are related by the enabling operator it means that after the completion of the first task, the second task will start. This means that the two tasks will belong to different enabled tasks sets. This aspect will be described by the changes that will affect the enabled task sets $\mathcal{E}$. In the following we will describe the inference rules for the enable operator:

$$\frac{[\![t_1]\!]\mathcal{E} \rightarrowtriangle [\![t_1']\!]\mathcal{E}}{[\![t_1 \gg t_2]\!]\mathcal{E} \rightarrowtriangle [\![t_1';\ t_2]\!]\ \mathcal{E}}\ (ER_1)$$

The first rule refers to the situation when the task t_1 is evaluated to one of its subtasks t_1'. In this case, the expression $[\![t_1 \gg t_2]\!]\mathcal{E}$ is evaluated to the expression $[\![t_1';\ t_2]\!]\ \mathcal{E}$. In this case the ETS remains unchanged, because further processing must be done.

$$\frac{[\![t_1]\!]\mathcal{E} \rightarrowtriangle [\![skip]\!]\mathcal{E} + \{t_1\}}{[\![t_1 \gg t_2]\!]\mathcal{E} \rightarrowtriangle [\![skip;\ t_2]\!]\ \mathcal{E} + \{t_1\}}\ (ER_2)$$

The second rule handles the situation when the enabling task t_1 is already accomplished (the next step in its performance is skip (no action)). In this case, the expression $[\![t_1 \gg t_2]\!]\mathcal{E}$ is evaluated to the sequential execution of skip followed by the execution of the enabled task t_2. The ETS will be enriched with the accomplished task t_1.

$$\frac{}{[\![skip \gg t_2]\!]\mathcal{E} \rightarrowtriangle [\![t_2]\!]\ \mathcal{E}}\ (ER_3)$$

The third rule describes the way the expression $[\![skip \gg t_2]\!]\mathcal{E}$ is evaluated. In this situation, only task t_2 must be executed and the ETS remains unchanged.

$$\frac{}{[\![t_1 \gg skip]\!]\mathcal{E} \rightarrowtriangle [\![t_1]\!]\ \mathcal{E}}\ (ER_4)$$

The fourth rule describes the situation when the enabling task is a final task (it is the last child of a node). In this case the expression $[\![t_1 \gg skip]\!]$ is evaluated to the execution of the enabling task.

3.2. **Choice operator semantics.** If two tasks t_1 and t_2 are related by the choice operator, the performance of the task depends on users option. That is why we have added o in the middle of the operator's notation, representing the users' option. We make the convention that if o evaluates to the constant 1, then the first task is selected for execution (see ChR_1). If o evaluates to 2, then the second task will be selected for execution (see ChR_3). In order to be able to select one of two options, both tasks should be available at the same time. This aspect is reflected on the updates which affect the store by adding both tasks to the ETS. In the following we will present the inference rules for the choice operator:

$$\frac{[\![o]\!]\mathcal{E} \twoheadrightarrow 1\ \mathcal{E} \qquad [\![t_1]\!]\mathcal{E} \twoheadrightarrow [\![skip]\!]\mathcal{E}'}{[\![t_1\ [o]\ t_2]\!]\mathcal{E} \twoheadrightarrow [\![skip]\!]\ \mathcal{E}' + \{t_1,\ t_2\}}\ (ChR_1)$$

$$\frac{[\![o]\!]\mathcal{E} \twoheadrightarrow 1\ \mathcal{E} \qquad [\![t_1]\!]\mathcal{E} \twoheadrightarrow [\![t_1']\!]\mathcal{E}'}{[\![t_1\ [o]\ t_2]\!]\mathcal{E} \twoheadrightarrow [\![t_1']\!]\ \mathcal{E}' + \{t_1',\ t_2\}}\ (ChR_2)$$

$$\frac{[\![o]\!]\mathcal{E} \twoheadrightarrow 2\ \mathcal{E} \qquad [\![t_2]\!]\mathcal{E} \twoheadrightarrow [\![skip]\!]\mathcal{E}'}{[\![t_1\ [o]\ t_2]\!]\mathcal{E} \twoheadrightarrow [\![skip]\!]\ \mathcal{E}' + \{t_1,\ t_2\}}\ (ChR_3)$$

$$\frac{[\![o]\!]\mathcal{E} \twoheadrightarrow 2\ \mathcal{E} \qquad [\![t_2]\!]\mathcal{E} \twoheadrightarrow [\![t_2']\!]\mathcal{E}'}{[\![t_1\ [o]\ t_2]\!]\mathcal{E} \twoheadrightarrow [\![t_2']\!]\ \mathcal{E}' + \{t_1,\ t_2'\}}\ (ChR_4)$$

3.3. **Disable operator semantics.** If t_1 and t_2 are two tasks and t_2 is a disabling task, this means that when t_2 performance starts t_1 is disabled no matter which is the execution state of t_1 (i.e. t_2 is able to suspend the execution of any subtask of t_1). That is why the rule does not have an hypothesis part of the inference rule. This means that t_2 must belong to every enabled tasks set generated by t_1. We have described this fact by updating the ETS $\mathcal{E}$ by adding the disabling task to each set of the ETS:

$$\frac{}{[\![t_1\ [>\ t_2]\!]\ \mathcal{E} \twoheadrightarrow [\![skip;\ t_2]\!]\ \mathcal{E}'}\ (\mathrm{DR})$$

where $\mathcal{E}' = \{\{l_1, \ldots, l_n, t_2\}\}$, $\forall\{l_1, \ldots, l_n\} \subset E$.

3.4. **Parallel composition operator semantics - Pure interleaving.** The pure interleaving operator relating the task T1 and T2 expresses nothing but any interleaving of the actions of T1 with the actions of T2:

$$\frac{}{[\![skip \ ||| \ skip]\!]\ \mathcal{E} \twoheadrightarrow [\![skip]\!]\ \mathcal{E}}\ (ConcR_1)$$

The first rule expresses the fact that by interleaving no actions (skip), the result will be also skip.

$$\frac{[\![t_1]\!]\mathcal{E} \twoheadrightarrow [\![t_1']\!]\mathcal{E}'}{[\![t_1 \ ||| \ skip]\!]\ M \twoheadrightarrow [\![t_1']\!]\ \mathcal{E}'}\ (ConcR_2)$$

$$\frac{[\![t_2]\!]\mathcal{E} \twoheadrightarrow [\![t_2']\!]\mathcal{E}'}{[\![skip \ ||| \ t_2]\!]\ M \twoheadrightarrow [\![t_2']\!]\ \mathcal{E}'}\ (ConcR_3)$$

The rules $ConcR_2$ and $ConcR_3$ describe the situation when one of the interleaving tasks is evaluated to one of its children and the ETS is changed to $\mathcal{E}'$, then interleaving the task with skip (no action), the result will be the execution of the child task and the ETS will be $\mathcal{E}'$.

$$\frac{[\![t_1]\!]\mathcal{E} \twoheadrightarrow [\![t_1']\!]\mathcal{E}'}{[\![t_1 \ ||| \ t_2]\!]\ \mathcal{E} \twoheadrightarrow [\![t_1' \ ||| \ t_2]\!]\ \mathcal{E}'}\ (ConcR_4)$$

$$\frac{[\![t_2]\!]\mathcal{E} \twoheadrightarrow [\![t_2']\!]\mathcal{E}'}{[\![t_1 \ ||| \ t_2]\!]\ \mathcal{E} \twoheadrightarrow [\![t_1 \ ||| \ t_2']\!]\ \mathcal{E}'}\ (ConcR_5)$$

$$\frac{\begin{array}{c}[\![t_1]\!]\mathcal{E} \twoheadrightarrow [\![t_1']\!]\mathcal{E}' \\ [\![t_2]\!]\mathcal{E} \twoheadrightarrow [\![t_2']\!]\mathcal{E}''\end{array}}{[\![t_1 \ ||| \ t_2]\!]\ \mathcal{E} \twoheadrightarrow [\![t_1' \ ||| \ t_2']\!]\ \mathcal{E}\mathcal{R}}\ (ConcR_6)$$

where $\mathcal{E}\mathcal{R}$ is the enabled tasks set formed as follows: $\forall\{X\} \subset \mathcal{E}', \forall\{Y\} \subset \mathcal{E}''$, $\mathcal{E}\mathcal{R} = \{\{X, Y\}\}$

The last three rules $ConcR_4$, $ConcR_5$ *and* $ConcR_6$ describe the situation when the execution of one (or both) of the interleaving tasks is evaluated to the execution of one of its children with changes on the ETS ($\mathcal{E}'$)(i.e. $[\![t_1]\!]\mathcal{E} \twoheadrightarrow [\![t_1']\!]\mathcal{E}'$). In this case, the result of the evaluation of interleaving of the two tasks will be evaluated to the interleaving of the child task(s) with implications on the ETS (i.e. $[\![t_1 \ ||| \ t_2]\!]\ \mathcal{E} \twoheadrightarrow [\![t_1' \ ||| \ t_2]\!]\ \mathcal{E}'$).

3.5. **Parallel composition operator semantics - Full Synchronization.** The rules for the parallel composition operator are very similar to those for the interleaving operator and will be presented in the following:

$$\frac{}{[\![\textbf{sync } t_1 \mid\mid \textbf{sync } t_2]\!]\ \mathcal{E} \twoheadrightarrow [\![t_1 \mid\mid t_2]\!]\ \mathcal{E}}\ (SyncR_1)$$

$$\frac{[\![t_2]\!]\mathcal{E} \twoheadrightarrow [\![t_2']\!]\mathcal{E}'}{[\![skip \mid\mid \textbf{sync } t_2]\!]\ \mathcal{E} \twoheadrightarrow [\![t_2]\!]\ \mathcal{E}'}\ (SyncR_2)$$

$$\frac{[\![t_1]\!]\mathcal{E} \twoheadrightarrow [\![t_1']\!]\mathcal{E}'}{[\![\textbf{sync } t_1 \mid\mid skip]\!]\ \mathcal{E} \twoheadrightarrow [\![t_1]\!]\ \mathcal{E}'}\ (SyncR_3)$$

$$\frac{[\![t_1]\!]\mathcal{E} \twoheadrightarrow [\![t_1']\!]\mathcal{E}'}{[\![t_1 \mid\mid \textbf{sync } t_2]\!]\ \mathcal{E} \twoheadrightarrow [\![t_1' \mid\mid \textbf{sync } t_2]\!]\ \mathcal{E}'}\ (SyncR_4)$$

$$\frac{[\![t_2]\!]\mathcal{E} \twoheadrightarrow [\![t_2']\!]\mathcal{E}'}{[\![\textbf{sync } t_1 \mid\mid t_2]\!]\ \mathcal{E} \twoheadrightarrow [\![\textbf{sync } t_1 \mid\mid t_2']\!]\ \mathcal{E}'}\ (SyncR_5)$$

$$\frac{\begin{array}{c}[\![t_1]\!]\mathcal{E} \twoheadrightarrow [\![t_1']\!]\mathcal{E}' \\ [\![t_2]\!]\mathcal{E} \twoheadrightarrow [\![t_2']\!]\mathcal{E}''\end{array}}{[\![\textbf{sync } t_1 \mid\mid \textbf{sync } t_2]\!]\ \mathcal{E} \twoheadrightarrow [\![t_1' \mid\mid t_2']\!]\ \mathcal{E}\mathcal{R}}\ (SyncR_6)$$

where $\mathcal{E}\mathcal{R}$ is the enabled tasks set formed as follows: $\forall\{X\} \subset \mathcal{E}', \forall\{Y\} \subset \mathcal{E}''$, $\mathcal{E}\mathcal{R} = \{\{X, Y\}\}$

3.6. **Order independence operator semantics.** The expression $t_1 \mid = \mid t_2$ means that t_1 and t_2 can be executed in any order, but both tasks must be executed (for example filling in the user name and the password in a login form). In terms of enabled task sets, both tasks will belong to the same enabled tasks set, fact illustrated by adding both tasks to the same enabled tasks set of ETS. The rules will be presented in the following:

$$\frac{[\![t_1]\!]\mathcal{E} \twoheadrightarrow [\![t_1']\!]\mathcal{E}}{[\![t_1 \mid = \mid t_2]\!]\ \mathcal{E} \twoheadrightarrow [\![t_1';\ t_2]\!]\ \mathcal{E} + \{t_1,\ t_2\}}\ (OIR_1)$$

$$\frac{[\![t_2]\!]\mathcal{E} \twoheadrightarrow [\![t_2']\!]\ \mathcal{E}}{[\![t_1 \mid = \mid t_2]\!]\mathcal{E} \twoheadrightarrow [\![t_1;\ t_2']\!]\ \mathcal{E} + \{t_2,\ t_1\}}\ (OIR_2)$$

The first two rules describe the situation when one of the two tasks is evaluated to the execution of one of its children (let's say t_1 is evaluated to the execution of t_1'). In this case, the expression $[\![t_1 \mid = \mid t_2]\!]\ \mathcal{E}$ is evaluated to the expression

$[\![t_1';\ t_2]\!]\ \mathcal{E} + \{t_1,\ t_2\}$ which means the execution of t_1' followed by the execution of t_2.

The following rules (OIR_3 and OIR_4) regard the situation when one of the tasks is skip and the other is evaluated to the execution of one of its subtasks. In this case the expression evaluates to the execution of the subtask and the ETS is updated by adding the parent task.

$$\frac{[\![t_2]\!]\mathcal{E} \rightarrow [\![t_2']\!]\ \mathcal{E}}{[\![skip \mid = \mid t_2]\!]\mathcal{E} \rightarrow [\![t_2']\!]\ \mathcal{E} + \{t_2\}}\ (OIR_3)$$

$$\frac{[\![t_1]\!]\mathcal{E} \rightarrow [\![t_1']\!]\ \mathcal{E}}{[\![t_1 \mid = \mid skip]\!]\mathcal{E} \rightarrow [\![t_1']\!]\ \mathcal{E} + \{t_1\}}\ (OIR_4)$$

3.7. **Iteration operator semantics.** The iteration operator associated to a task means that the task is iterative (after completing an execution, the task can start its execution again).

The first inference rule say that if a task execution is accomplished ($[\![t_1]\!]\mathcal{E} \rightarrow [\![skip]\!]\mathcal{E}$) then task's iteration is evaluated to skip and the ETS is updated by adding a new set containing the iterative task:

$$\frac{[\![t_1]\!]\mathcal{E} \rightarrow [\![skip]\!]\mathcal{E}}{[\![t_1^*]\!]\ \mathcal{E} \rightarrow [\![skip]\!]\ \mathcal{E} + \{t_1\}}\ (ItR_1)$$

If the iterative task is not accomplished (one of its subtasks is running), then the iteration of the task is evaluated to the iteration of its child and the ETS is updated by adding the parent task.

$$\frac{[\![t_1]\!]\mathcal{E} \rightarrow [\![t_1']\!]\mathcal{E} + \{t_1\}}{[\![t_1^*]\!]\ \mathcal{E} \rightarrow [\![t_1'^*]\!]\ \mathcal{E} + \{t_1\}}\ (ItR_2)$$

3.8. **Complementary rules.** In addition to the rules associated to each operator we will need some rules for sequential execution. The need for these rules appears when composed temporal operators should be handled. The evaluation of such kind of expressions is reduced to the evaluation of some sequential tasks. The rules are presented in the following:

$$\frac{}{[\![skip; t]\!]\ \mathcal{E} \rightarrow [\![t]\!]\ \mathcal{E}}\ (SR_1)$$

The rule SR_1 says that skip (no action) followed by the execution of the task t is evaluated to the execution of task t and the ETS is not affected.

$$\frac{[\![t_1]\!]\mathcal{E} \rightarrow [\![t_1']\!]\mathcal{E}'}{[\![t_1; t_2]\!]\ \mathcal{E} \rightarrow [\![t_1'; t_2]\!]\ \mathcal{E}'}\ (SR_2)$$

If the task t_1 is evaluated to the execution of one of its children t_1', the expression $[\![t_1; t_2]\!]\ \mathcal{E}$ is evaluated to the execution of the child task (t_1') followed by the execution of t_2. The ETS is updated with the changes introduced by the execution of t_1'.

$$\frac{t_1 \rightarrow skip}{[\![t_1]\!]\ \mathcal{E} \rightarrow [\![skip]\!]\ \mathcal{E} + \{t_1\}}\ (ExR_1)$$

The rule ExR_1 is used when t_1 is a leaf in the task tree and its execution is completed (is transformed in skip). In this case the expression $[\![t_1]\!]\ \mathcal{E}$ where $\mathcal{E}$ is the ETS is evaluated to skip and the ETS is updated with a new task (t_1).

4. Examples

In the following we will present some examples of generating enabled tasks sets using the operational semantics. The first example is related to a task tree using the enabling operator. The task tree is presented in Figure 1. We have to evaluate the expression $[\![B \gg D \gg E]\!]\mathcal{E}$ which is the frontier of the tree. Using the rule

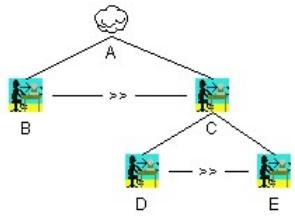

Figure 1. Task tree with enabling operators

ER_2 we will make the first step in our deduction as follows.

$$\frac{[\![B]\!]\mathcal{E} \rightarrow [\![skip]\!]\mathcal{E} + \{B\}}{[\![B \gg D \gg E]\!]\mathcal{E} \rightarrow [\![skip;\ D \gg E]\!]\ \mathcal{E} + \{B\}}$$

Let us denote $\mathcal{E}' = \mathcal{E} + \{B\}$. By applying the rule SR_1 we will evaluate $[\![skip;\ D \gg E]\!]\mathcal{E}'$.

$$\frac{}{[\![skip;\ D \gg E]\!]\ \mathcal{E}' \rightarrow [\![D \gg E]\!]\ \mathcal{E}'}$$

Now, using ER_2 we have:

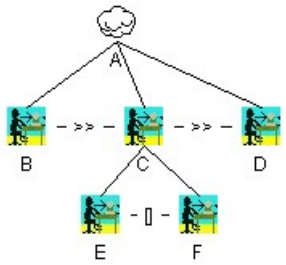

FIGURE 2. Task tree with choice operator

$$\frac{[\![D]\!]\mathcal{E}' \rightarrow [\![skip]\!]\mathcal{E}' + \{D\}}{[\![D \gg E]\!]\mathcal{E}' \rightarrow [\![skip;\ E]\!]\ \mathcal{E}' + \{D\}}$$

Let us denote $\mathcal{E}'' = \mathcal{E}' + \{D\}$. Using The rule SR_1, $[\![skip; E]\!]\mathcal{E}''$ is evaluated to $[\![E]\!]\ \mathcal{E}''$.

$$\frac{}{[\![skip; E]\!]\mathcal{E}'' \rightarrow [\![E]\!]\ \mathcal{E}''}$$

Now, we will apply ExR_1 to evaluate $[\![E]\!]\ \mathcal{E}''$ and we will obtain:

$$\frac{E \rightarrow skip}{[\![E]\!]\mathcal{E}'' \rightarrow [\![skip]\!]\ \mathcal{E}'' + \{E\}}$$

At the end of the deduction process the final content of the store will be composed by three enabled tasks sets: $\{B\}$, $\{D\}$, and $\{E\}$.

The second example handles a task tree using the choice operator. We will consider that the user selects the first option from the available ones (see Figure 2). We have to evaluate the expression $[\![B \gg E[o]F \gg D]\!]\mathcal{E}$. Using ER_2 for the task B we will obtain:

$$\frac{[\![B]\!]\mathcal{E} \rightarrow [\![skip]\!]\ \mathcal{E} + \{B\}}{[\![B \gg E\ [o]\ F \gg D]\!]\mathcal{E} \rightarrow [\![skip;\ E\ [o]\ F \gg D]\!]\ \mathcal{E} + \{B\}}$$

Let us denote $\mathcal{E}' = \mathcal{E} + \{B\}$. Applying ChR_1 and ER_2we will obtain:

$$\frac{\dfrac{[\![o]\!]\mathcal{E}' \rightarrow\ 1\ \mathcal{E}' \qquad E \rightarrow skip}{[\![E\ [o]\ F]\!]\mathcal{E}' \rightarrow [\![skip]\!]\mathcal{E}' + \{E, F\}}}{[\![E\ [o]\ F \gg D]\!]\mathcal{E}' \rightarrow [\![skip; D]\!]\mathcal{E}' + \{E, F\}}$$

Let us denote $\mathcal{E}'' = \mathcal{E}' + \{E, F\}$. We will apply SR_1 and the next step in the deductive process will be:

$$\frac{}{[\![skip; D]\!]\mathcal{E}'' \rightarrow [\![D]\!]\mathcal{E}''}$$

Applying ExR_1 we will obtain:

$$\frac{D \rightarrow skip}{[\![D]\!]\mathcal{E}'' \rightarrow [\![skip]\!]\mathcal{E}'' + \{D\}}$$

At the end of the deductive process the content of the ETS is $\mathcal{E}'' + \{D\}$ which means $\{E, F\},\ \{D\}, \{B\}$.

The following example illustrates the process of building ETS when the task model contains parallel (interleaving) tasks. The expression we must evaluate is $D \gg E[]F \,|||\, G[]H$(see Figure 3).

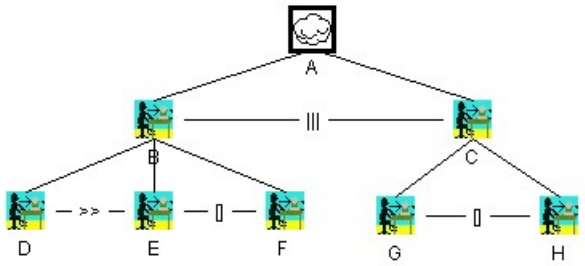

FIGURE 3. Task tree using the pure interleaving operator

$$\frac{[\![D]\!]\mathcal{E} \rightarrow [\![skip]\!]\mathcal{E} + \{D\}}{[\![D \ \gg\ E\ [o_1]\ F \,|||\, G\ [o_2]\ H]\!]\mathcal{E} \rightarrow [\![skip;\ E\ [o_1]\ F \,|||\, G\ [o_2]\ H]\!]\ \mathcal{E} + \{D\}}$$

Let us denote $\mathcal{E}' = \mathcal{E} + \{D\}$. We will evaluate $[\![skip;\ E\ [o_1]\ F \,|||\, G\ [o_2]\ H]\!]\ \mathcal{E}'$ using SR_1.

$$\frac{}{[\![skip;\ E\ [o_1]\ F \,|||\, G\ [o_2]\ H]\!]\mathcal{E}' \rightarrow [\![E\ [o_1]\ F \,|||\, G\ [o_2]\ H]\!]\ \mathcal{E}'}$$

Applying SR_1 and ChR_1 we will obtain:

$$\frac{\dfrac{[\![o_1]\!]\ \mathcal{E}' \qquad E \rightarrow skip}{[\![E\ [o_1]\ F]\!]\ \mathcal{E}' \rightarrow [\![skip]\!]\mathcal{E}' + \{E, F\}}}{[\![E\ [o_1]\ F \,|||\, G\ [o_2]\ H]\!]\ \mathcal{E}' \rightarrow [\![skip\ \,|||\, G\ [o_2]\ H]\!]\ \mathcal{E}' + \{E, F\}}$$

Let us denote $\mathcal{E}'' = \mathcal{E}' + \{E, F\}$. Using $ConcR_3$ and ChR_2 the result will be:

$$\frac{\dfrac{[\![o_2]\!]\ \mathcal{E}'' \rightarrow 2\ \mathcal{E}'' + \{G, H\} \qquad G \rightarrow skip}{[\![G\ [o_2]\ H]\!]\mathcal{E}'' \rightarrow [\![skip]\!]\mathcal{E}'' + \{G, H\}}}{[\![skip\ \,|||\, G\ [o_2]\ H]\!]\ \mathcal{E}'' \rightarrow [\![skip \,|||\, skip]\!]\ \mathcal{E}\mathcal{R}}$$

The final ETS will be $\mathcal{E}\mathcal{R} = \{\{D, G, H\},\ \{E, F, G, H\}$.

5. Conclusions and further work

In this article we have presented a new approach of task models based on operational semantics. We have given a definition of task trees and we have written inference rules for the temporal operators used to describe task models. We have captured on the inference rules aspects regarding the updates that take place at

the enabled tasks sets collection ($\mathcal{E}$). The process of building the presentation model will be based on the obtained ETS collection.

As future research directions our goals are: to build the presentation model based on the enabled tasks sets obtained using operational semantics and to extend our approach in order to build a dialog model also (transitions between states) starting from task models.

References

[1] J. Annett and K.D. Duncan. Task analysis and training design. *Journal of Occupational Psychology*, 41:211–221, 1967.

[2] T. Bolognesi and E. Brinksma. Introduction to the iso specification language lotos. *Comput. Netw. ISDN Syst.*, 14(1):25–59, 1987.

[3] S. Card, T. Moran, and A. Newell. *The Psychology of Human-Computer Interaction*. Cariere. Lawrence Erlbaum Associates, 1983.

[4] L. Marucci, F. Paterno, and C. Santoro. *Multiple and Cross-Platform User Interfaces: Engineering and Application Frameworks*, chapter Supporting Interactions with Heterogeneous Platforms Through User and Task Models, pages 217–238. H. Javahery and A. Seffah (eds.), 2003.

[5] G. Mori, F. Paternò, and C. Santoro. CTTE: Support for developing and analyzing task models for interactive system design. *IEEE Transactions on Software Engineering*, 28(9):1–17, 2002.

[6] NIST. National institute for standardization and technology. http://www.nist.gov/dads/HTML/tree.html.

[7] F. Paternò. Model-based tools for pervasive usability. *Interacting with Computers*, 2004.

[8] F. Paternò, C. Mancini, and S. Meniconi. ConcurTaskTrees: A diagrammatic notation for specifying task models. In *INTERACT '97: Proceedings of the IFIP TC13 Interantional Conference on Human-Computer Interaction*, pages 362–369. Chapman & Hall, Ltd., 1997.

[9] S.J. Payne. Task Action Grammar. In Bullinger H. J. and Shackel B., editors, *Proceedings INTERACT'84*, pages 139–144, North-Holland, 1984.

[10] G. D. Plotkin. A Structural Approach to Operational Semantics. Technical Report DAIMI FN-19, University of Aarhus, 1981.

[11] D. Scapin and C. Pierret-Golbreich. Towards a method for task description: MAD. *Work with Display Units*, 89:371–380, 1989.

[12] B. Sufrin and R. Bornat. Animating Operational Semantics with JAPE. citeseer.ist.psu.edu/485702.html.

[13] R. van Loo, G. van der Veer, and M. van Welie. Groupware Task Analysis in practice: a scientific approach meets security problems. In *7th European Conference on Cognitive Science Approaches to Process Control*, 1999.

[14] M. van Welie. *Task-based User Interface Design*. PhD thesis, Vrije Universiteit Amsterdam, 2001.

[15] M. van Welie, G. van der Veer, and A. Koster. Integrated representations for task modeling. In *Tenth European Conference on Cognitive Ergonomics*, pages 129–138, 21-23 August 2000.

[16] M. van Welie and G.C. van der Veer. Structured methods and creativity: a happy Dutch marriage. In *Co-Designing 2000*, 2000.

Department of Computer Science, Babeş-Bolyai University, Cluj-Napoca, Romania
E-mail address: adriana, motogna@cs.ubbcluj.ro