FORMAL DESCRIPTION OF HUMAN MACHINE (H-M) INTERACTIONS DERIVED FROM THE APPLICATION TASK GRAPH

Guy Camilleri

IRIT
Université Paul Sabatier,118 route de Narbonne 31062 Toulouse Cedex France
Camiller@irit.fr

Abstract :
To design interactive systems, a clear description of the interactions is required. A deep understanding of H-M interactions is necessary in order to formulate a useful description. The task graph (TG) structure and therefore the different tasks, which could be achieved by the system, appears essential for interaction specifications.

In this contribution, we propose a generic formalism, which can represent H-M interactions from any TG by determining constraints that it entails on the interaction. A formal and generic definition of the task, recipe and task graph concepts is required to represent generated interactions by any application. The interaction description depends on projected functionalities by an application and the way they are achieved. It is supplied by TG features. The tasks distribution (agents which perform them) and their realisation control (order of tasks performance) are two TG criteria which we determine as influencing interaction. Different methods can be applied for each TG type (which are defined with the two previous criteria) to specify the interaction. The utilisation of tasks which modify the TG, allows interactions to be described with the same formalism (task, recipe, TG and methods) out of any TG type of applicative domain (containing prescribed tasks).

Keywords: Human machine interaction, task, transaction, Interaction formalisation.

1 Introduction

To design interactive systems, a clear description of the interactions is required. A deep understanding of H-M interactions is necessary in order to formulate a useful description. The task graph (TG) structure and therefore the different tasks, which could be achieved by the system, appears essential for interaction specifications [Sucrow, 1997]. TG represents the hierarchical decomposition of task performance. An example of the TG for the performance of a phone task is “pick up the phone”, then “dial the number” which is composed of “research the number” then “dial it”, then “speak”, and “hang up”.

In this contribution, we propose a formalism, which can represent H-M interactions from any TG. The TG is always built during the system design phase. Generally, this graph is implicit, but it is partially and often dynamically visible through the system Human Machine Interface (HMI). Throughout interactions, users build a mental TG, which represents the task hierarchy achievable with system. Therefore, one main purpose of the HMI is to allow the user, during the interactions, to build a faithful representation of parts of TG.

A formal description of interactions derived from the TG depends on the graph nature and the formalism used to represent it. In the literature, the same formalism is often used for the TG and the interaction specification. A strong relationship exists between the TG and the interactions, the latter of which are derived from the TG for a given application. This is because the interactions depend on some application functionalities (prescribed tasks) and the way they are achieved (some parts of TG).

First, we define the concepts used to formalise the TG. Then we expose the features and the different TG types, which influence the interaction definition. Finally, we provide a simple formalism to describe interactions.

2 Task Graph definition

The TG definition following the applications often varies [Isténes, 1996]. Therefore, we propose an abstract TG definition which can be used in any application domain. We also choose a simple formalism to represent the concepts, which are easily expandable and customisable to match with the particular application needs.

2.1 Task

The task notion is used in several domains, often as primitive (axiom), therefore it is difficult to find a common definition. For experts systems, a task denotes an instance of a problem, a problem class, and both a problem class and an abstract description of a method of solving the problem [Chandrasekan and Johnson, 1992]. In knowledge acquisition, following the application domain and the acquisition method used, the task definition varies. For example, In KADS method [Schreiber et al 1993], a task is a composite action of problem solving entailing subtask decomposition, moreover a task can perform a particular goal. However, For MACAO method [Soubie, 1996], a task is an autonomous entity of treatment which allows to reach a goal from an input set of elements. In Fiona application a task is a concrete action, while in DSTM framework a task represents an objective to reach (a goal) [Isténes, 1996]. The task concept is also present in ergonomics. They mainly consider two task kinds, the prescribed and effective ones [De Montmollin, 1995]. The effective tasks are effectively achieved by an operator, by his activity. The main difference between prescribe tasks and effective tasks is: the prescribed tasks are defined by the designer, while the effective tasks are really performed by an operator.

In all these definitions, the action concept appears as a common notion (work to do, action to achieve, action of problem solving, and so on). Therefore, we propose below a task definition based on the action concept.

Definition: A task is an action in particular context, which is performed to reach a goal to satisfy a given intention.

Note: In the task definition, the intention term designates a forward-directed intention. Forward-directed intentions defined by Bratman in [Bratman, 1990] represent the agent desire to perform tasks not achieved but for which he is committed.

Formal definition: A task is composed of:

- Task's type
- Parameters list (p_1, …,p_n) with $n \in \mathbf{N}$

Example: Phone(Jean,05-61-61-61-61,10h30) is a task where Phone is the task's type and (Jean,05-61-61-61-61,10h30) is the parameter list.

Logical formalisation: Tasks are represented by logical terms [Grosz and Kraus, 1996]. Then, they denote an element of interpretation domain. To access to the different parameters, we use the role functions, which are described, in our logical language by some function terms. For example, the role function agent(T) returns the agent which performs T.

2.2 Recipe

The recipe concept presented here is similar to the Grosz's recipe concept [Grosz and Kraus, 1996].

Definition: A recipe describes a task performance. It represents the hierarchical decomposition (at one abstraction level) of a task performance under appropriate constraints.

For example, a recipe for the performance of a phone task is "pick up the phone", and then "dial the number", and then "speak", and "hang up".

Formal definition: A recipe for the task T is a triplet made up of T, a subtask set $\{T_{11}; \ldots;T_{1n}\}$ and a constraint set $\{C_1; \ldots;C_m\}$, noted $(T,\{T_{11}; \ldots;T_{1n}\},\{C_1; \ldots;C_m\})$.

- Example: A recipe for the performance of phone task is (Phone(people,number, time), {pick_up_phone(T_1) ;Dial_number(number,T_2) ;Speak(T_3); Hang_up(T_4) }, $\{T_1<T_2<T_3<T_4, \ldots\}$).
- Graphical representation:

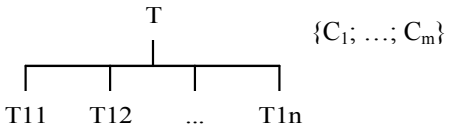

Figure 1: Recipe for task T

Logical formalisation: A complete and detailed presentation of the logical formalisation of recipe concept can be found in [Camilleri, 1999]. To represent the performed task, we use the DO operator introduced by Grosz [Grosz and Kraus, 1996].

A recipe for a task A composed of subtasks {B1; …;Bn} $n \in \mathbf{N}$, under the constraints Γ is represented by a logical entailment (or recipe axiom) as :

$\forall x\ DO(B_1(s_1(x))) \wedge DO(B_2(s_2(x))) \wedge \ldots \wedge DO(B_n(s_n(x))) \wedge \Gamma \supset DO(A(x))$

Where A,B1, …,Bn represent predicates of task type and $s_1(x)$, …, $s_n(x)$ describe some decomposition role functions and,Γ conjunctive constraint set.

2.3 Task Graph

The task graph notion can be viewed as a kind of recipe library [Grosz and Kraus, 1996], or as a reasoning model which is a part of conceptual model used in knowledge acquisition.

Definition: A Task Graph is a collection of recipes hierarchically organised. It represents all possible decompositions (following the described recipes) of tasks performance.

- Graphical representation:

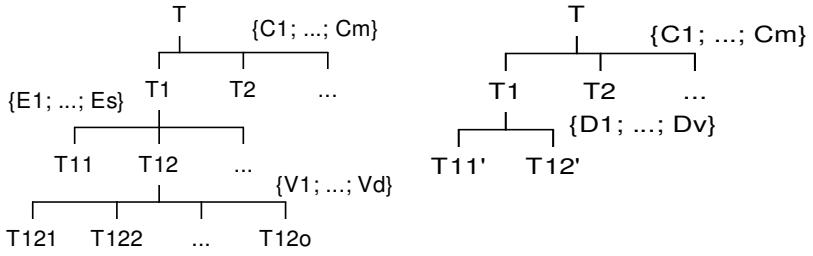

Figure 2: Task Graph

In the figure 2, the sets {C1; …;Cm}, … represent the constraints set of their respective recipes. For example, a TG for the performance of a phone task is “pick up the phone”, then “dial the number” which is composed of “research the number” then “dial it”, then “speak”, and “hang up”.

Logical formalisation: A TG is represented by a conjunctive set of recipe axioms.

3 TG's features influencing interaction

We identified two characteristics (tasks performance control and tasks distribution), which are important to define the interactions needed from the TG. To describe the interaction, we use the following notions [Post, 1996]: transactions which concern the transfer tasks of information ingredients between agents; transaction plans, which define the performance order of transactions.

3.1 Tasks performance control

Generally, the transactional plans follow the control structure (tasks performance order) of the TG. In the above example (TG for the performance of a phone task), suppose that each task (pick up, dial, speak, hang up) needs a transaction, the hang up transaction completion requires the speak transaction achievement and so on. Therefore, transactions are achieved in the same order as the tasks performance.

3.2 Tasks distribution

Tasks distribution represents all possible associations between agents and tasks. It can be modeled by doublets (agent, task) set. The tasks distribution is an important criteria to define the tasks needing a transaction. For example, consider the following TG (reduced here to a single recipe): (calculating((x+y-z-t)/2,R,T),{add(x,y,r1,T_1); sub_1(r1,z,r2,T_2); sub_2(r2, t,r3,T_3);div(r3,2,R,T_4)}, {$T_1<T_2<T_3<T_4$; $T_1 \subset T$ …}). For d1={(user,add); (user,sub1); (user,sub2); (system,div)} and d2={(user,add); (system,sub1); (user, sub2); (system, div)} distributions, the transactions required are different. In the d1 distribution only a transaction that presents the outcome r3 (sub2 result) is needed. For the d2 distribution, four transactions are required to present the intermediate task result to the other agents. Thus, the possible set of distributions define tasks which require a transaction (and the nature of these transactions).

4 Different TG types

A control is called static if and only if it is predefined (explicitly described in the TG), likewise a distribution is static if and only if the tasks distribution is predefined. We consider the following TG's types:

- Type 1 (Static Control (SC), Static Distribution (SD)) during the conception process, tasks requiring transactions are known (SD) in the same way as their achievement order (SC). We can define completely the required transaction plans. Therefore, all transactional plans are determined to the conception stage. Transactions can be bound to the TG. In [Sucrow, 1997], this TG's property is used to bind widgets to user goals.
- Type 2 (Dynamic Control (DC), SD) throughout the system conception, all transactions can be defined (SD). As we don't know the application order of tasks (DC), so the transaction order (thus transactional plans) cannot be determined. However, to achieve TG's tasks, the task control must be defined during the TG performance. This definition can require an explicit transaction or be determined by another task achievement.

- Type 3 (CS, Dynamic Distribution (DD)) at the conception stage, tasks needing transactions are not known (DD). As for the Type 2 TG, the distribution can be defined during the execution by an explicit transaction or by another task achievement.
- Type 4 (DC, DD) during the conception, tasks requiring a transaction (DD) and the execution order (DC) are not known. Therefore, during the TG execution these two features must be fixed either by some specific transactions or by tasks performance.

5 Transaction plan formalisation

We represent a transaction by a task in our model. Therefore, a transaction plan is modelled by a TG containing transactional tasks.

This way to formalise transaction plans allows to unify the data manipulation mechanisms (TG is the only structure used) and to merge the domain TG with the interaction aspects of this TG. Moreover, the merged TG makes an explicit relationship between transaction tasks and related domain tasks. The method proposed above (to treat the type 2, 3, 4 TG) is modelled in this formalism by a transaction task, which modifies the TG. Thus, we need to construct another TG, which represents the system state.

6 Conclusion

In this paper, we propose a generic formalism allowing to describe H-M interactions in any applicative domain from the TG by determining the constraints, which generate on interactions. The main advantages come from using the same formalism to represent the TG and interactions. It facilitates their links and unifies the data manipulation mechanisms (TG is the only structure used).

The dissociation of the interaction description and its representation is an important point of interactive system conception [Camilleri, 1998]. Indeed, it clarifies the design and simplifies the system maintenance. A possible method to undertake the dissociation is to create an autonomous agent dedicated to interaction materialisation. Further work will attempt to use the link between the transaction tasks and the domain tasks to recognise user intentions.

References

[Bratman, 1990] Bratman, M. E. (1990). What is intentions?. In MIT Press, P.R Cohen, J.L. Morgan, M.E. Pollack, Intentions in communication, pages 15-31, Cambridge, MA, USA.

[Camilleri, 1998] Camilleri, G. (1998). Description formelle de l'interaction Homme Machine. Internal Report 98-33-R, IRIT, Toulouse, France.

[Camilleri, 1999] Camilleri, G. (1999). A formal theory of plan recognition adapted to discourse systems. Internal Report 99-16-R, IRIT, Toulouse, France.

[Chandrasekan and Johnson, 1992] Chandrasekaran, B. and Johnson, T. (1992). Generic Tasks and Task structures: History, Critique and New Directions. In Springer-Verlag, J-M David, J-P Krivine, R Simmons, Second Generation Expert Systems, ISBN 3-540-56192-7, pages 232-272.

[De Montmollin, 1995] De Montmollin, M. (1995). What is intentions?. In MIT Press, P.R Cohen, J.L. Morgan, M.E. Pollack, Intentions in communication, pages 15-31, Cambridge, MA, USA.

[Grosz and Kraus, 1996] Grosz, B. and Kraus, S. (1996). Collaborative Plans for complex group action. *Journal of Artificial Intelligence*, 86(2): 269-357.

[Isténes, 1996] Isténes, Z. (1996). Zola a language to operationalise Conceptual Models of Reasoning. *Journal of Computing and information*, 2(1): 689-706.

[Post, 1996] Post, W.M. (1996). Cooperative emergency call-handling: Design and Realisation. COOP'96, pages 83-103

[Schreiber et al, 1993] Schreiber, G. Wielinga, B. Breuker, J. (1993). Modelling Expertise. In Academic Press Harcourt Brace Jovanovich, G. Schreiber, B. Wielinga, J. Breuker, A principled approach to knowledge-based system development, ISBN 0-12-629010-7, pages 22-91.

[Soubie, 1996] Soubie, J-L. (1996). Coopération et Système à base de Connaissance, Mémoire d'habilitation, UPS-Toulouse III, IRIT, France.

[Sucrow, 1997] Sucrow, B. (1997). Formal Specification of Human-Computer Interaction by Graph Grammars under Consideration of Information Resources. In *Proceedings* of the 1997 Automated Software Engineering Conference (ASEC'97), pages 28-35.