Tailoring the Presentation of Plans to Users' Knowledge and Capabilites

Detlef Küpper[1] and Alfred Kobsa[2]

[1] University of Applied Science Aalen, Beethovenstr. 1, 73433 Aalen, Germany
detlef.kuepper@fh-aalen.de
[2] Dept. of Information and Computer Science, University of California, Irvine, CA 92697-3425, U.S.A.
kobsa@ics.uci.edu

Abstract. Tailoring advice to a user means finding a plan by which she can reach her goal, and supplying the missing knowledge that she needs to successfully execute the plan. The paper presents a method to determine the kind and amount of this missing knowledge for an already generated domain plan. We show that both a user's knowledge and his capabilities to perform actions must be taken into account when deciding on a plan presentation that is suitable for him. We also argue that it may be useful to consider issues of plan presentation already during the planning process, and show how this can be accomplished in a planning system.

1 Introduction

The effectiveness of advice-giving systems essentially hinges on users' ability to take advantage of given advice to reach their goals. The problem of whether a user can *understand* advice has already received considerable attention in the research literature (see, e.g. [19] for a survey). However, the user must also be able to execute the advice, i.e. have the *capabilities* to perform each step of the advice. Capability in this sense means the user's personal abilities and her authorization to perform the actions that occur in the advice.

Küpper and Kobsa [9] presented a plan generation approach for achieving the user's goals that considers the user's capabilities. The resulting plans are then *in principle* executable by the user. In order to perform the plan, she may however need additional information. In this paper, we describe our approach to identify this missing user knowledge, starting with a plan that the user can in principle execute and that is suitable for reaching her goals. We will show that the additional knowledge that must still be communicated depends not only on the user's existing knowledge but again also on her capabilities.

In the remainder of this paper, we will first summarize central characteristics of the user model that we employ and introduce some terminology. The next section discusses the different types of knowledge that users must possess to be able to perform a plan. We then present an algorithm that determines this knowledge for a plan generated for the current user. This algorithm also calculates the "presentation cost"

of the plan to be communicated, which is proportional to the structural complexity of the presentation and can be regarded as a coarse estimate of the user's comprehension efforts. Afterwards, we re-integrate our results into the plan generation process to ascertain that plans will preferably be generated that the user is not only capable of performing but also have the lowest presentation costs. We also discuss other measures besides presentation cost for rating plan presentations, which take user skills, user preferences, and the likelihood of success into account. Finally we describe some related research, and outline future work to decide which parts of the missing knowledge identified by our algorithm should be presented explicitly or may be omitted, depending on inferences that the user may draw.

2 Elements of our Approach

For representing assumptions about the user, we employ an enhanced version of the user model in [9], which separates system assumptions about users' knowledge and their capabilities. User capabilities are modeled a*s plan operators* that a user is in principle able and authorized to perform. They have a terminological representation in the user model, the so-called *plan concepts*. Their preconditions and effects are represented by attributes of their plan concepts. Plan operators/concepts can be instantiated for the generation of user-tailored plans, and become the *steps* of the plans. Plan concepts are also used for modeling (the system's assumptions about) the user's planning knowledge, and the system's own planning knowledge. Since plan steps are instances of plan concepts, assumptions about the user's knowledge about plan steps are inherited from the assumptions about his knowledge about plan concepts. The same holds true for user capabilities.

Each plan concept can also be associated with one or more *decompositions* in the models of the system's and the user's knowledge. Such decompositions are sets of partially ordered plan steps and thus (simplified) plans. Steps of decompositions are again instantiations of plan concepts. This is roughly comparable to decompositions in hierarchical planning [15,□3], except that plan concepts in our decompositions usually do not possess preconditions and effects and cannot be used for planning any more. Instead, decompositions are canned recipes of how to perform the associated plan concept. This is justified since we require the plan generation process that precedes plan presentation to consider all relevant conditions and dependencies.

Küpper and Kobsa [9] describe this planning process for generating plans that the user is able to perform. While its exact nature is irrelevant for presentation purposes (we use UCPOP [12]), we require that the resulting plan not only contain steps but also *causal links*, which are by-products of the planning process. A causal link (*s*,□*l*,□*r*) is a relation between a step *s*, a condition represented by literal *l* that is satisfied by *s*, and a step *r* that requires *l* for its executability. The recognition of user's capabilities and plan knowledge is not directly part of this work since it is highly domain-dependent (see [13,□9,□14] for relevant work).

3 Identifying the Contents for Plan Presentation

Plan presentation should enable users to perform the generated plan and thereby reach their goals. A plan for a user should not be decomposed to the lowest possible level (e.g., down to finger movements). This would considerably increase the communication and comprehension efforts, and lower users' acceptance of the resulting lengthy advice (from a technical point of view, the domain modeling effort and the planning complexity would also soar). Instead, plan presentation should rely on users' ability to further elaborate received plans [21]. Such plans still include, e.g., abstract plan concepts that need to be refined, and abstract object descriptions that need to be replaced by identified objects. When such a plan becomes presented to the user, she is expected to continue decomposing plan steps into a sequence of more primitive executable actions (which may even be well beyond the system's domain model). If unforeseen obstacles hinder the successful execution of a more abstract plan, the user still has a chance to modify it. Besides a listing of the plan steps in correct order, the user therefore needs knowledge about the *properties* of and *interrelations* between the plan steps, and she must know how to *perform*, i.e. further decompose, the steps of the plan.[1]

3.1 Knowing Plan Steps and Their Interrelations

Information about the role of each step in a plan is provided by the causal links of the plan. They describe what step satisfies which precondition for what other step. This knowledge is important for modifying the plan when unexpected obstacles occur, since each modification of the plan must take care not to threaten a causal link. If a plan contains a causal link (s, l, r), no modifications between step s and step r are allowed that negate l since otherwise the precondition of step r is not satisfied anymore and the plan will fail.

3.2 Knowing How to Perform a Step of a Plan

Users may need additional advice for decomposing plan steps if their knowledge is insufficient or even wrong. A plan concept is marked as *atomic* in the user model if the user does not need further information about it (such plan concepts may be directly executable, or the user may have extensive competence for further plan decomposition that goes beyond the system's domain model). For non-atomic plan concepts, the decision on what to explain to the user will hinge on her presumed knowledge how to execute the plan concept. Such knowledge is represented by the presence of one or more decompositions of the plan concept in the user's knowledge model. But even when the user has such knowledge, it may be wrong, or not usable since the user is unable to perform it. Therefore we provide explanations of all

[1] Furthermore, she may require referential descriptions and properties of objects that play a role in the plan [16, 17].

presented plan concepts except when the user's knowledge about how to perform the concept is assumed to be *reliable*. We call a plan concept *reliable* if

a) none of its decompositions in the user's knowledge model is false, and
b) at least one of them is *usable single-handedly* by the user.

The first condition aims at preventing that the user will consider wrong execution alternatives. The reference point for judging correctness is the system's domain model: a decomposition of a plan concept in the user's knowledge model is correct only if it is also part of the system's knowledge. Assumptions about such user misconceptions may stem from knowledge about typical errors of certain user groups, or from the recognition of individual user misconceptions [14]. A decomposition is usable *single-handedly* (by the current user) if he can in principle perform all its steps and has reliable knowledge about how to perform them if they are non-atomic. Note that the steps of a decomposition may in turn be instantiations of non-atomic plan concepts which the user must know how to perform as well (in the domains that we investigated so far, the nesting was found to be fairly shallow though).

For plan concepts that are not reliable (and which therefore require further explanation of how they can be executed), the system must select one of the plan's decompositions from its own knowledge. Different decompositions generally do not lead to equally good explanations (more on this in the next subsection), and some of them may not even be usable by the current user. We call a decomposition *usable* (by the current user) if he can in principle carry out all its steps and, for each non-atomic step, the system knows at least one decomposition that is usable by the current user. For an example, consider the plan concept *shutdown-windows-computer* with the following two decompositions:

(1) push keys <Ctrl>, <Alt> and <Del> simultaneously,
    then select <shutdown-button>;
(2) select <Start> from the task bar, then <Shutdown>.

Although both recipes will shut down a computer under Windows, the former will not be usable for someone who cannot hit three keys simultaneously.

If the system knows several usable decompositions of a plan concept, it should select the "best". The presentation algorithm described below computes a valuation of each available alternative. The rating of an alternative is a function of the ratings of the components of its decomposition and, if the components are plan concepts that require further explanation, the valuation of the decompositions that will be used for these explanations.

3.3 The Presentation Algorithm

The pseudo code of Fig. 1 summarizes the algorithm for determining the knowledge that a user needs to successfully execute a plan *p*. The task is distributed among three functions which return two values, namely a data structure that represents the contents of the presentation, and a numerical value *cost* that represents the rating of this presentation. We use the auxiliary functions *presentation* and *cost* to access these two results.

present-plan (p:plan, cs:set-of-plan-concepts)
(p1) set *cost* initially to (| *steps* (*p*) | + 2 * | *causal-links* (*p*) |) * k_{pres}
(p2) set *presentation* initially to *linearize* (*p*)
(p3) for each *s* ∈ *steps* (*p*)
add the name of *plan-concept* (*s*) to *s* in *presentation*
(p4) for each *pc* ∈ *plan-concepts* (*p*) \ *cs*
(p5) add results of *present-plan-concept* (*pc*, *cs*) to *presentation*, *cost*
(p6) add *pc* and all *plan-concepts* of
presentation (*present-plan-concept* (*pc*, *cs*)) to *cs*

present-plan-concept (*pc*:plan-concept, *cs*:set-of-plan-concepts)
(c1) set *presentation* initially to {}
set *cost* initially to 0
(c2) if user doesn't know *pc*
(c3) set *presentation* to descript (*pc*)
set *cost* to *cost*(descript (*pc*))
(c4) if not (*pc* is atomic or user has reliable knowledge of *pc*)
(c5) add results of present-recipe-for-plan-concept (*pc*, *cs*) to
presentation, *cost*

present-recipe-for-plan-concept (*pc*:plan-concept, *cs*:set-of-plan-concepts)
(r1) let *ec* be the set of usable decompositions of *pc* according to system's knowledge
(r2) if *ec* = {} ⇒ set *cost* to (plan presentation fails)
(r3) else set *presentation*, *cost* to results of *present-plan* (*d*, *cs*) where
(r4) *d* ∈ *ec* ∧
cost (*present-plan* (*d*, *cs*)) = $\min_{(d' \in ec)}$ *cost*(*present-plan* (*d'*, *cs*))

Fig. 1. Pseudo code of the plan presentation algorithm

The function *present-plan* processes generated plans as well as decompositions of plan concepts. It starts with a linearization[2] of the plan (p2) and adds the name of the corresponding plan concept for each plan step (p3). This should enable the user to identify the steps. Function *present-plan-concept* determines whether the user needs information on plan concepts, and what kind of information. The function is called for each plan concept unless it had already been processed (the bookkeeping parameter *cs* will be described below). An explanation of a plan concept may consist of a canned description if the user does not know the concept (c2,c3), and user-specific information about how to perform it if the plan concept is non-atomic and the user does not have reliable knowledge (c4,c5). User-specific information about how to perform a plan concept is determined by the function *present-recipe-for-plan-concept*. It selects the decomposition with the minimal cost (r4) from the set of usable decompositions of the plan concept (r1). The presentations of these decompositions are recursively computed by the function *present-plan*. Line (r2) handles the case in which the system

[2] The plan generation process may result in a plan that is only partially ordered.

cannot give an explanation that is *usable* by the current user. The subsequent enhanced plan generation process excludes such *unexplainable* plan concepts from further consideration and thus produces plans only for which the algorithm can come up with an adequate presentation.

Let us now discuss some more details of the presentation algorithm. We use the auxiliary functions *steps*, *causal-links* and *plan-concepts,* which return the set of steps, causal-links and plan concepts of a plan, respectively. The function *plan-concept* of a plan step yields the plan concept of which the step is an instantiation.

To avoid multiple explanations of the same concept, the loop starting at line (p4) excludes plan concepts that are in *cs*, the set of already processed plan concepts. This set is passed on in all subsequent function calls as their second argument. Since an explanation of a plan concept may contain explanations of other plan concepts that are also used later in the plan, the set is updated after each call of *present-plan-concept* (p6).

Besides its main task of determining the missing knowledge that the user needs for successful plan execution, the algorithm also computes the *presentation-cost*. Each component of the presentation contributes to this value: steps and causal links in line (p1), and descriptions of plan concepts in line (c3). Following the propositional theory of text comprehension [5,6], this may be seen as a rough estimation of the user's effort to comprehend the presentation since all components carry new information that must be processed by the user if they are presented explicitly. In (p1), we use the factor k_{pres} as a unit for the presentation cost. If we use it to estimate the user's comprehension effort, k_{pres} stands for the effort that is caused by the presentation of one plan step. Since causal links are more complex components, each contributes two units to the presentation cost.

3.4 An Example

This example demonstrates the behavior of our algorithm for different system assumptions about users' knowledge and capabilities. Users download files through FTP by first selecting the file and then initiating the transfer. Fig. 2 shows three possible decompositions of the corresponding plan concept *ftp-dnload-bin*:

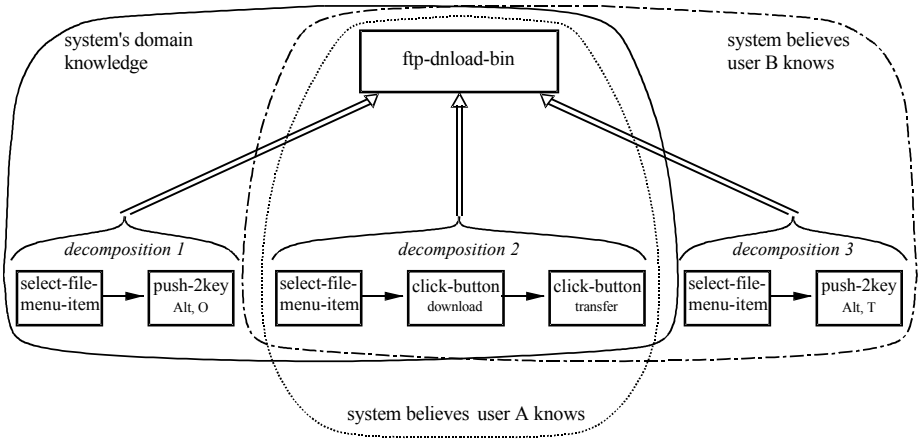

Fig. 2. Different planning knowledge of the system and two users

1. *select-file-menu-item* and *push-2key (Alt,O):*
   users select the file and hit <*Alt*>-<*O*> to start the transfer.
2. *select-file-menu-item*, *click-button (download)* and *click-button (transfer):*
   users select the file, chose the transfer direction, and initiate the transfer.
3. *select-file-menu-item* and *push-2key(Alt,T):*
   same as (1), except <*Alt*>-<*T*> is hit instead of <*Alt*>-<*O*>.

(1) and (2) are correct decompositions of *ftp-dnload-bin* while (3) models a frequently occurring mistake. The former are therefore included in the system's domain model while the latter is not (see Figure 2). Assume that user A knows decomposition 2, is able to perform all its steps, and that all steps are atomic for him. The respective plan concepts are therefore contained both in the knowledge and the capabilities part of A's user model, and marked as atomic in the knowledge part. User A therefore has reliable knowledge about *ftp-dnload-bin* and does not need further explanations thereon. Function *present-plan-concept* in line (p5) of Fig. 1 yields an empty presentation for this plan-concept, and 0 for the *presentation-cost*.

Assume that user B believes that *ftp-dnload-bin* can be decomposed into decompositions 2 and 3. Both are therefore contained in the knowledge part of B's user model. From the system's point of view, B believes in both correct and incorrect decompositions. Since her beliefs about *ftp-dnload-bin* are thus not reliable, the system must provide its own decomposition. To this end, function *present-plan-concept* calls *present-recipe-for-plan-concept* (c5), which must decide between the presentation of decompositions 1 or 2. If B is able to perform all steps of either decomposition and if all are atomic, both decompositions are *usable*. The decision between them in (r3,r4) is made by applying *present-plan* to both alternatives and opting for the one with the lower returned presentation costs. In (p1-p3), a presentation is constructed consisting of two steps for decomposition 1 and three steps for decomposition 2, respectively. The costs are calculated as $2 * \Delta_{pres}$ and $3 * \Delta_{pres}$, (lines (p4-p6) add no extra costs since all plan steps are atomic), and decomposition 1 is thus preferred in (r3,r4).

If user B would not know how to perform *push-2key (Alt, O)*, then *present-plan-concept* (p5) would generate a decomposition of this concept as well (via recursive calls of *present-recipe-for-plan-concept* and *present-plan*), and add its costs to 2 * k_{pres}. The total *presentation-cost* of decomposition 2 will be lower in this case. Decomposition 2 will also be selected if the user is not able to perform *push-2key (Alt, O)* since, say, these keys are relatively far apart. In this case, decomposition 1 would not be in the set of usable decompositions and hence is not considered any further.

4 Plan Valuation During Plan Generation

Up to now, our discussion separated the two phases of user-tailored advice (i.e., plan generation and plan presentation), in order to work out more clearly the processes and knowledge sources that support them. A generation process that completely disregards the need for subsequent plan presentation may however end up with a plan that requires tedious explanations or, in the worst case, does not have an acceptable presentation at all.

We can identify two starting points for considering presentation aspects during plan generation without curtailing the solution space of the presentation component. First, we exclude unexplainable non-atomic plan concepts from the planning process, because their occurrence in a plan always leads to a rejection of the presentation. The second onset exploits a planning control feature in UCPOP. Gerevini and Schubert [4] investigate several search control strategies to speed up the planning process. We use some of their results to bias the planning process to prefer "good" plans, i.e. plans for which the presentation algorithm will compute a low presentation-cost. The UCPOP algorithm works on several plan candidates in parallel. Before each processing step, it rates all candidates and selects the candidate with the best rating to process next. We use a rating function that considers the presentation-cost of the plan candidate. UCPOP will therefore first operate on the plan candidate with the lowest presentation-cost. If UCPOP adds steps or causal links to the plan candidate, the presentation-cost of this candidate increases and UCPOP may work on another candidate in the next step if it has a better rating. The first solution found has the lowest presentation-cost. This means that plans with a low presentation-cost are preferred, but no solution is lost. Similar control mechanisms are available in most practical plan generators.

For efficiency reasons we do not use the function *present-plan* to calculate the presentation-cost. Instead, we compute the *direct presentation-cost* and the *explanation set* of each plan concept before the planning process starts. The *direct presentation-cost* of a plan concept is determined by the cost of its description if the user does not know the concept, and by the number of (direct) steps of an explanation if the user does not know how to perform it. The *explanation set* of a plan concept is the set of plan concepts that would be (indirectly) used in an explanation of how to perform the plan concept if the user would need such an explanation, and the empty set otherwise. Thus the rating of a plan candidate *p* can be computed as the sum of

a) (| *steps*(*p*) | + 2 * | *causal-links* (*p*) |) * k_{pres} plus

b) the direct presentation-cost of each plan concept $pc \in$ (*plan-concepts*(p) $\setminus$ $\bigcup_{pc' \in plan\text{-}concepts(p)}$ *explanation-set*(pc')).

This value is a close approximation of the value that would be computed by the function *present-plan*. However, the plan concepts for a plan in the presentation algorithm may influence the decision between alternative explanations of plan concepts that are used later in the plan. This may result in a lower presentation-cost than the value computed for the plan candidate.

We use this technique of biasing the planning process in the way described above only, but also see three other applications scenarios in user-tailored plan presentation:

1. Users may prefer performing certain plan concepts over other plan concepts.
2. Users may have more practice performing certain plan concepts than others [13].
3. Certain plan concepts may have a lower risk of failure for certain users.

User preferences, user practice and likelihood of failure can be expressed by appropriate cost factors that will be taken into account in the planning process. These cost factors can be assigned to the respective plan concept in the capability part of the user model. In contrast to presentation-cost, they relate to the execution of a plan, and not its presentation or comprehension. The recursive combination function for determining these costs therefore will have to consider the complete plan, i.e. not only those plan structures that become explicitly communicated to the user but also those that the user will still expand.

5 Related Work

An early piece of work on plan presentation was the system of Mellish and Evans [11] that produced a natural language description of plans generated by the planner NONLIN [18]. However, this approach neglected the users' needs and characteristics, and usually yielded descriptions that contained too much detail to be useful for user advice. Subsequent work on the generation of instructional text [10,13] and multimedia presentation [20,1] considered user characteristics, but did not take users' capabilities to perform plan steps into account nor a further elaboration of the plan by the user. Young's approach [22] is closely related to ours in that he also generated descriptions from the result of a planning process. He focused on curtailing descriptions by removing information that the user can easily infer, but did not take individual differences in the planning knowledge of users into account.

While our approach tries to identify the missing knowledge that users need for successful plan execution, other research [19] focuses on the surface structure of the presentation. The goal in this case is that users should be able to comprehend the presentation, i.e. be able to incorporate all intended information into their internal knowledge structures. We view these approaches as complementary to ours since the knowledge determined by our algorithm still needs to be passed on to the user in an adequate communicative manner.

Other related research includes negotiations with the user about the system's proposal, like in collaborative dialog systems [2]. In this approach, a user may reject advice of the system if he believes it is inadequate. This may especially be the case if the user has wrong assumptions about how to perform plan steps.

6 Summary and Further Development

This work dealt with the problem of determining the missing knowledge that the current user needs for being able to successfully execute an already generated domain plan. Additionally we demonstrated that it may be useful to consider issues of plan presentation during the planning process already, and showed how this can be accomplished in a planning system.

Our approach was guided by the idea that plan descriptions for human users are generally not immediately executable, but need further elaboration by the recipient. This requires a description that enables the user to comprehend not only the sequence of steps that must be executed, but also the internal structure and functionality of the plan. Further knowledge is required if the user does not know, or has wrong assumptions about, how to carry out some plan steps. We showed the knowledge that is required for these tasks, and analyzed how properties of the user influence the kind and amount of this knowledge. Particularly the presentation of knowledge about how to perform plan steps benefited from a user model that separates the user's knowledge from his capabilities. However, even the best plan from a plan generation point of view is useless if there is no good way to explain it to the user. This motivated the improvement of the planning process by taking presentation issues into account.

Additional work is required to generate presentations that are adequate for a user. One important question is, what subset of the knowledge to be communicated (as identified by our algorithm) should be explicitly presented to the user, since presenting the complete knowledge usually leads to lengthy descriptions. Young [22] showed that users make fewer errors during plan execution if the plan description does not contain information that they can easily infer. He proposed to omit such information from a plan presentation. For deciding which plan components should be omitted, he simulated the user's inferences by a user-specific planning algorithm. Currently, we also work on this problem. Our approach is similar to Young's, in that it is also based on the assumption that a user will continue the planning process when she obtains an incomplete plan from the system. However, we do not postulate a user-specific planning algorithm but rather believe that the user's planning *knowledge* has the most impact on his ability to infer missing plan components. The decision whether or not a component of the plan should be presented explicitly is made on the basis of a comparison between the user's effort to comprehend and memorize a description of this component and the his effort to infer it. Details of this work can be found in [8, ch. 4.4].

Whereas most assumptions in our model are based on studies in (text and discourse) comprehension (e.g., [6, 7]), a new line of research will be to find empirical answers through experiments to a number of questions that remained open so far. For instance, in the example of Section 3.4 the system decided to explain the shortest but unknown decomposition, rather than referring to the longer but known decomposition, or even correcting the user's misconception once and forever. We foresee user experiments in environments where users' plan knowledge and abilities can be highly controlled (e.g., computer games with first-person player) to determine additional factors besides explanation length that may influence the choice between these explanatory options.

7 Implementation

A prototype of the system described here has been implemented in CommonLisp and is available from http://linux2.image.fh-aalen.de/kuepper/Diss-CD/.

References

1. André, E.: *A plan-based approach to the generation of multi-modal presentations* (in German). INFIX-Verlag, Sankt Augustin, Germany (1995)
2. Chu-Carroll, J. and Carberry, S.: Collaborative response generation in planning dialogues. *Computational Linguistics 24*(3) (1998) 355-400
3. Erol, K.; Hendler, J. and Nau, D.: Semantics for hierarchical task-network planning. Techn. Report CS-TR-3239, Computer Science Dept., Univ. of Maryland (1994)
4. Gerevini, A. and Schubert, L.K.: Accelerating partial-order planners: Some techniques for effective search control and pruning. *Journal of Artificial Intelligence Research 5* (1996) 95-137
5. Kintsch, W. and van Dijk, T.A.: Toward a model of text comprehension and production. *Psychological Review 85* (1978) 363-394
6. Kintsch, W. and Vipond, D.: Reading comprehension and readability in educational practice and psychological theory. In *Perspectives on memory research*, Nilsson, L.-G. ed., Erlbaum, Hillsdale, NJ (1979) 329-365
7. Kintsch, W.: *Comprehension: A Paradigm for Cognition.* Cambridge University Press, Cambridge (1998)
8. Küpper, D.: *User modeling for user-tailored plan generation and presentation* (in German). Academic Publishing Corporation / IOS Press, Berlin, Amsterdam, forthcoming (2003)
9. Küpper, D. and Kobsa, A.: User-tailored plan generation. In *User Modeling: Proc. of the 7th International Conference, UM'99*. Springer, Wien (1999) 45-54
10. McKeown, K.R.; Elhadad, M.; Fukumoto, Y.; Lim, J.; Lombardi, C.; Robin, J. and Smadja, F.: Natural language generation in COMET. In *Current Research in Natural Language Generation*, Dale, R.; Mellish, C. and Zock, M. eds., Academic Press, London, UK (1990)
11. Mellish, C. and Evans, R.: Natural language generation from plans. *Computational Linguistics 15*(4) (1989) 233-249
12. Penberthy, J.S. and Weld, D.S.: UCPOP: A sound, complete, partial order planner for ADL. In *Principles of Knowledge Representation and Reasoning: Proc. of 3rd International Conference (KR'92)*. Morgan Kaufmann, San Mateo, CA (1992) 103-114
13. Peter, G. and Rösner, D.: User-model-driven generation of instructions. *User Modeling and User-Adapted Interaction 3*(4) (1994) 289-319
14. Pollack, M.E.: Plans as complex mental attitudes. In *Intentions in Communication*, Cohen, P.R.; Morgan, J.; Pollack, M.E., MIT Press, Cambridge, MA (1990) 77-103
15. Sacerdoti, E.D.: *A Structure for Plans and Behavior*. Elsevier/North-Holland, Amsterdam (1977)
16. Sarner, M. and Carberry, S.: Generating tailored definitions using a multifaceted user model. *User Modeling and User-Adapted Interaction 2*(3) (1992) 181-210
17. Schmauks, D. and Reithinger, N.: Generating multimodal output: Conditions, advantages and problems. In *Proc. of 12th International Conference on Computational Linguistics*. North-Holland, Amsterdam (1988) 584-588
18. Tate, A.: Project planning using a hierarchic non-linear planner. Research Report 25, Univ. of Edinburgh, Department of Artificial Intelligence (1976)
19. van Mulken, S.: *User Modeling for Multimedia Interfaces: Studies in Text and Graphics*

Understanding. Dt. Univ.-Verlag, Wiesbaden, Germany (1999)
20. Wahlster, W.; André, E.; Finkler, W.; Profitlich, H.-J. and Rist, T.: Plan-based integration of natural language and graphics generation. *Artificial Intelligence 63* (1993) 387-427
21. Webber, B.; Badler, N.; Di Eugenio, B.; Geib, C.; Levison, L. and Moore, M.: Instructions, Intentions and Expectations. *Artificial Intelligence 73*(1-2) (1995) 253-269
22. Young, R.M.: Using Grice's maxim of quantity to select the content of plan descriptions. *Artificial Intelligence 115*(2) (1999) 215-256