A Dialogue Approach for Solving Wicked Planning Problems

Extended Abstract

Gengshen Du
University of Calgary
2500 University Drive NW
Calgary, AB, T2N 1N4, Canada
dug@cpsc.ucalgary.ca

Michael M. Richter
TU Kaiserslautern
FB Informatik, P.O.Box 3049
67653 Kaiserslautern, Germany
richter@informatik.uni-kl.de

Guenther Ruhe
University of Calgary
2500 University Drive NW
Calgary, AB, T2N 1N4, Canada
ruhe@ucalgary.ca

1. Introduction

Although wicked problems don’t have a precise definition, they have several characteristics [Rittel, Webber 1984]:

- There is no definite formulation of a wicked problem.
- There is no stopping rule and therefore one can always improve the solution.
- Solutions are not true-or-false but good-or-bad.
- The problem is not static but changes dynamically.
- The view on problems and their solutions is subjective and context dependent. One has different participants (called stakeholders) with different preferences and this makes it problematic to judge the quality of the solution.
- Every problem is essentially unique and this makes it difficult to make use of experiences.

In this paper we consider an interactive and explanation supported approach to planning problems that are not only wicked but also complex. As a consequence, humans as well as software agents, mostly optimization procedures, are involved. In our approach we concentrate on the communications between the agents. These are organized in the form of a dialogue between the agents that has an explanatory character.

This explanation method differs essentially from the methods used in traditional expert systems. During the dialogue stakeholder opinions can be changed or withdrawn, even if they are formulated as hard constraints (i.e. the constraints that traditionally can not change).

This method is generic to be applied to any wicked and complex planning problems. Therefore, we also present the applications of the method in release planning and urban planning.

2. Problem Description

Because of the nature of the wicked planning problems, its modeling is an important but difficult issue. In order to get computer support for modeling, one has to map all concepts, relations and intentions of the real situation to a mathematical model. This mapping is based on the hope that the planner will generate a useful or at least acceptable plan. To some degree this is only a guess because if the planning problem contains the difficulties described above one cannot anticipate in which way the planner will use the parameter values in order to generate the plan. In addition, certain real world aspects may not be covered at all. In such a situation two problems arise:

- The user often has no insight in the major reasons that lead to the presented solution. In particular, if the solution is somewhat surprising she/he has difficulties to accept it. Therefore the user may not trust and may not accept the solution.
- Because of the lack of understanding, the user is not able to give a feedback.

One can see that in interactive planning the generation of some plan is not the end of the planning, it is rather considered as a first step. Solving the above problems calls for a deep human involvement in the modeling as well as in the problem solving. This would necessitate tool support. As a result, we get a situation where human and software agents cooperate and explanations are necessary.

3. Background of Explanations

Explanations play an essential role in interactive processes between agents. An overview is given in [Wooley 1998]. They are always answers to questions, may they be raised or not. The fundamental approach from the logic of question and answer is given in [Belnap, Steel 1976].

Here we restricts ourselves to explanations in the context of problem solving and decision making, in particular to the “Why” and “Why not” questions of the form Q = Why did you do/not do X? We introduce a new classification of explanation that is intuitively understandable and also useful for implementation issues. When dealing with wicked problems these methods may be combined.

- Backward explanations

Backward explanations refer to something that happened in the past. The (backward) answer to Q is: Because you forced me to do/not do X. The purpose of backward explanation is to increase the acceptance of solution. Besides this cognitive science view there is a pragmatic computer science view. For providing such an answer the constraints and rules have to be stated explicitly and the search for them has to be sufficiently efficient. In many classical knowledge based systems this was the case and
backwards explanation was popular by following the parsing paradigm. This was, however, not possible for, e.g., procedural programs.

- Forward explanations
  Forward explanations look into the future. The (forward) answer to Q is: If I would have done/not done X then the following unwanted consequence Y would have occurred. The forward explanation can, on the one hand, also increase the acceptance of the solution but it can also be used to improve and complete the solution itself. The reason is that they can be employed during the interactive problem formulation and solution process. The technical problem discussed here is to provide forward explanations that are both useful and easy to overlook and handle for plan improvement.

4. The Generic Approach

4.1 Overview

We consider a wicked planning problem and looking for an ideal solution from the user’s point of view. The generic approach proposed in this paper is called EXPLAIN-DIALOG. It performs a dialog between an intelligent system (planner) and the user based on the following assumptions:

- The planner presents several solution alternatives to the user.
- The user is able to define an ideal solution.
- Allows to locally modify plans towards a defined ideal solution.

The main steps of the approach are:

1. Generate of one or more solution alternatives.
2. Formulate the stakeholders’ opinion in a compact way called concern C which indicates the objective of the planning problem.
3. Present an ideal solution for each concern, called prototype prot(C).
4. Compare the prototype and the actual solution by a similarity measure simC. In order to simplify the situation the system shows a reduced form of the actual plan to the user that just contains the attributes that are relevant and where the prototype and the actual plan differ significantly.
5. Allow the user to propose changes in the solution and to remove or weaken normative constraints which are the decisions made by humans.
6. Generate a new plan and repeat the process. The presentation of the new plan is a forward explanation.
7. The user may now react in three different ways: either prefer the new plan or still accept the old plan because the reasons are now understood better or choose another concern and iterate the process.

The main aspects of the proposed approach are:

- Reducing the complexity:
  - By focusing the attention of the user to a specific view at a time.
  - By simplifying the situation using qualitative descriptions.
  - Showing major qualitative consequences of proposed plan changes from different concerns.

4.2 Characteristics

It is important that different kinds of applications applying this generic approach have only few undefined and application dependent terms, such as the concerns and the similarity measures.

A plan is represented as plan = (p, i ∈ I) and PLANS is the set of plans. The concerns form an abstract set and each concern C we assume the existence of some prototype prot(C) with the intention that the concern is realized in an ideal way.

Similarity measures compare plans:

simC: PLANS x PLANS → [0, 1]

The similarity measure as well as the prototype may be user dependent. The similarity measures are formulated as weighted sums with a weight vector w = (w_1, ..., w_n) of non-negative coefficients:

simC((a_1, ..., a_n), (b_1, ..., b_n)) = (w_i × (a_i, b_i) | 1 ≤ i ≤ n)

where a_i and b_i (1 ≤ i ≤ n) represent a given plan and the ideal solution, respectively.

The following functions on the basis of the similarity measures that are essential for the dialogue approach:

- DangerDegree(C, plan) = 1 − simC(plan, prot(C))
- QualitativeDanger(C, plan) is introduced by taking three qualitative regions 0 ≤ α < β < 1 in which the function takes the values low, medium and high, respectively.
- The predicate DangerC is defined by:
  DangerC(plan) ↔ QualitativeDangerC(plan) = high
- The function computeConsequences takes as input the values of some solution variables and the hard constraints. It generates the logical consequences of fixed values for the variables under the constraints.
- The reduced comparison of plan_1 and plan_2 is:
  plan_red(plan_1, plan_2, sim) = (plan_1, plan_2, sim) = ((plan_1, plan_2)) ∈ I, simC(plan_1, plan_2) < s, w_i > t). Given a concern C the reduced representation of a plan p is obtained by comparison with the prototype:
  plan_red(p, C) = (p, i ∈ I, simC(p, prot(C)) < s, w_i > t)

The thresholds s and t are user-determined. This means the reduced plan shows those attributes that are the major reasons for the deviation from the prototype.

5. Instantiation: Release Planning

Release planning is conducted in early stage of software development to generate release plans. A release plan arranges the requirements (the tasks) R = {R_1, ..., R_n} developed for (a) large software product in a sequence. This sequence is cut into releases where each release contains a maximum number of requirements so that the
the total effort of the segment fit into the effort allowed for a release, all the technology constraints are met, and stakeholder satisfaction is achieved to the maximum extent [Ruhe 2005].

It also has to consider the following hard constraints [Ruhe, Ngo-The 2004]:

- Precedence constraints between requirements $R_i$ and $R_j$ that specify $R_i$ must be implemented before $R_j$
- Coupling constraints between requirements $R_i$ and $R_j$ that specify they must be implemented in the same release
- Resource constraints that indicate the available capacity for each release and the needed capacity for each requirement
- Pre-assignments that fix the release of a requirement

In addition, weak constraints are supposed to reflect opinions and preferences of different stakeholders involved in terms of votes. The votes are concerned with the priority of delivering requirements and value of requirements.

Release planning is a problem that has the previously mentioned characteristics of the wicked planning problems. Specifically, its difficulties are [Ruhe, Saliu 2004]:

- Many aspects, including even the objectives are not stated precisely and in a formal way and are in addition context dependent.
- There are different stakeholders who have diverging interests.
- There are uncertain estimates concerning efforts (this is a standard problem in software development).
- Personal opinions about importance or urgency that may be overruled in a specific context.

For the instance of release planning, the generic explanation approach introduced in Section 4 can be applied. In [Du 2004], the formal approach was preliminary implemented for release planning using ReleasePlanner$^*$ as a release planning tool (the system). Figure 1 shows an instantiated procedure for this dialogue based on the generic procedure.

Another example to be discussed is more complex and comes from urban planning. It is in the spirit of [Rittel, Webber 1984]. Presently the dialog in this application is done as indicated but by humans only and the computer support is under investigation.

6. Conclusion

We presented a dialogue based approach to improve solutions of the wicked planning problems. It was based on the observation of the characteristics of wicked problems. The dialogue has an explanatory character that gives some insight into decisions so that the user may change previous opinions or may stay to it. The proposed generic approach aims at increasing user trust on the wicked planning results. This approach can be instantiated and applied to many applications, e.g. release planning and urban planning.

![Figure 1: Instantiated DIALOG-EXPLAIN procedure for release planning](image)

References


