C2O: A Tool for Guided Decision-Making

Alexander Nöhrer
Institute for Systems Engineering and Automation
Johannes Kepler University Linz, Austria
alexander.noehrer@jku.at

Alexander Egyed
Institute for Systems Engineering and Automation
Johannes Kepler University Linz, Austria
alexander.egyed@jku.at

ABSTRACT

Decision models are widely used in software engineering to describe and restrict decision-making (e.g., deriving a product from a product-line). Since decisions are typically interdependent, conflicts during decision-making are inevitably reached when invalid combinations of decisions are made. Unfortunately, the current state-of-the-art provides little support for dealing with such conflicts. On the one hand, some conflicts can be avoided by providing more freedom in which order decisions are made (i.e., most important decisions first). On the other hand, conflicts are unavoidable at times and living with conflicts may be preferable over forcing the user to fix them right away – particularly, because fixing conflicts becomes easier the more is known about an user's intentions. This paper introduces the C2O (Configurator 2.0) tool for guided decision-making. The tool allows the user to answer questions in an arbitrary order – with and without the presence of conflicts. While giving users those freedoms, it still supports and guides them by 1) rearranging the order of questions according to their potential to minimize user input, 2) providing guidance to avoid follow-on conflicts, and 3) supporting users in fixing conflicts at a later time.

Categories and Subject Descriptors: I.2.8 Problem Solving, Control Methods, and Search: Heuristic methods

General Terms: Human Factors, Verification.

1. INTRODUCTION

In software engineering, many tasks exist that involve making decisions [3, 5]. This paper introduces a tool for guided decision-making – focusing on the product-line domain. From a decision oriented view, a decision model consists of questions and choices for these questions, which are typically interrelated. For example, an online car configurator is such a decision model where the car manufacturer offers a predefined set of car types with predefined features. Relations typically impose or restrict choices. For example, choosing a convertible makes asking about roof racks irrelevant for obvious reasons. Not all relations are that obvious.

During decision-making, decisions may thus lead to conflicts if they violate such relations. It is quite easy to detect such conflicts (through SAT-Solvers [2] or consistency checkers [4]). It is also quite easy to prevent conflicts by disabling choices that would lead to conflicts. Doing so is state-of-the-practice, however this is problematic whenever the decision maker reaches a point where a desired choice is no longer available – a dead end. Typically, tool support to avoid and deal with dead ends is rarely existent and when a dead end is reached, decision makers often have no choice but to backtrack and start over. We identified the following main reasons for dead ends:

1. Decision makers are required to answer questions first that may not be important to them. This unnecessarily restricts the answers to follow-on questions.

2. Decision makers are unaware of the consequences of their choices because they do not understand all relations, which can result in favored choices for unaswered questions being excluded.

The first issue stems from the fact that a predefined order for answering questions restricts the user's freedom in answering questions [7]. This issue is largely ignored in literature but of essential importance because users' needs vary and users prefer to answer questions first that are important to them or easy to answer – before being led through the remaining questions. Since the user may not know how to answer the remaining questions first, such that favored choices are not eliminated, such a freedom could help avoid possible dead ends significantly. For example, most online car configurators "force" the user to first select the type of car before selecting other properties (e.g., engine, color). However, what if the user cares more about mileage than make? Any prewired solution, even if optimal, that deviates from the user's preferred way of answering the questions then forces the user to exhaustively explore all choices to find the ones that satisfy the desired properties. While it would be simple to allow users to answer questions arbitrarily, there are good reasons to restrict the order. A pre-defined order can be optimized to avoid unnecessary questions to be answered. The second issue is a matter of visualization which some tools already address.

Naturally, even by providing the freedom to answer questions arbitrarily, doing so may not prevent dead ends altogether. Existing tool support for decision-making does not allow conflicts and as such it forces the user to backtrack decisions to resolve the dead end before continuing. Unfortunately, such backtracking is quite complex and users may prefer to continue making decisions, even accepting the resulting conflicts (i.e., tolerating inconsistencies [1]). The freedom to select any choice during decision-making, even a conflicting one, is a legitimate way forward without forcing

published at
ASE'10, September 20–24, 2010, Antwerp, Belgium.

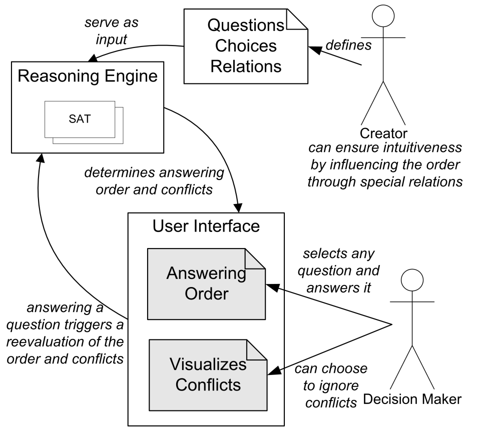

serve as input
Questions Choices Relations
defines
Reasoning Engine
SAT
Creator
can ensure intuitiveness by influencing the order through special relations
determines answering order and conflicts
User Interface
Answering Order
selects any question and answers it
answering a question triggers a reevaluation of the order and conflicts
Visualizes Conflicts
can choose to ignore conflicts
Decision Maker

Figure 1: Overview of the system's internals

the user to deal with a problem that may not be important from a user's perspective at that time.

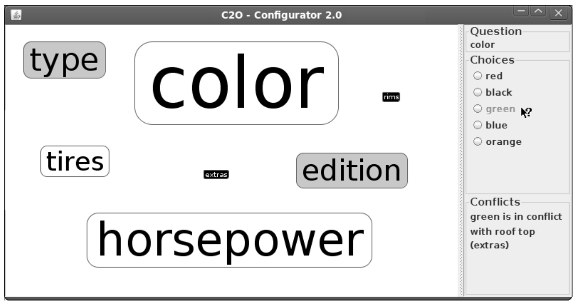
This paper introduces a product configurator tool that allows users to answer questions and introduce conflicts arbitrarily – but provides the necessary capabilities to support the user during decision-making: Firstly users can make decisions in any arbitrary order. However guidance is still provided: the questions are presented in an optimal (with regard to minimizing user input) order, which the user can choose to ignore at any time. Secondly consequences of decisions can be visualized beforehand if the user wishes. And finally our tool allows the configuration process at a dead end to be continued by entering a conflicting state. However users will be guided through the remaining questions to avoid follow-on conflicts and any additional answers provided will be used later to help resolve the conflict. Our tool incorporates strategies described elsewhere [6]. The following summarizes these capabilities.

2. TOOL AND ARCHITECTURE

The C2O tool, a prototype, currently puts emphasis on the necessary reasoning behind guided decision-making. Figure 1 depicts a basic overview of the system's internals and interfaces to the users (creator and decision maker). Currently only the interaction between the decision maker and the system has a graphical user interface, defining questions, choices and relations is done programmatically.

We use a SAT-Solver as our main reasoning engine. Feature Models, Product-lines or general Constraint Satisfaction Problems (CSP) are transformed into Conjunctive Normal Form (CNF) and fed into the SAT-Solver. The SAT-Solver then serves as an oracle to answer questions about the impact of decisions. Before the decision-making process starts, the initial optimal order of questions is determined with the help of a heuristic . During the decision-making process this order is reevaluated after every decision and the results are communicated to the user. For the creators of decision models this implies savings in not having to predefine the optimal order which is exponentially complex. For the decision maker this implies more freedom in answering questions while still benefiting from optimizations.

In addition we incorporated our approach of tolerating conflicts [6] into the tool. It visualizes conflicting choices, provides explanations on conflicts, and allows conflicting decisions. Most importantly of all, it lets the user proceed in the presence of conflicts, even encourages the continuation to gather more information about the source of the conflict (defect), to resolve it later.



green is in conflict with roof top (extras)

Figure 2: Excerpt of using C2O to configure a car

Both these technologies have been excessively evaluated. We have demonstrated that our approach for determining an ideal order is 92-100% optimal and automatically reduces up to a third of all questions compared to a random selection. We also found that at the time a conflict is discovered, it is around 29-71% likely that there is more than one option for fixing it. Yet, the longer conflicts are tolerated the less complex becomes their fixing (i.e., because decisions not only restrict choices for answering future questions but also choices for fixing conflicts).

Figure 2 shows an excerpt of our tool that shows some questions, their sizes are directly related to their potential for minimizing user input if answered. User decisions are indicated with a black background (they are also the smallest ones since they already have been answered), conflicting user decisions have a gray background, and unanswered questions have a white background.

Acknowledgments

The authors would like to thank Thomas Schartmüller for his work on the GUI. This research was funded by the Austrian FWF under agreement P21321-N15.

3. REFERENCES

[1] R. Balzer. Tolerating Inconsistency. In *ICSE*, pages 158–165, 1991.
[2] M. Davis, G. Logemann, and D. W. Loveland. A machine program for theorem-proving. *Commun. ACM*, 5(7):394–397, 1962.
[3] D. Dhungana, R. Rabiser, P. Grünbacher, K. Lehner, and C. Federspiel. DOPLER: An Adaptable Tool Suite for Product Line Engineering. In *SPLC (2)*, pages 151–152. Kindai Kagaku Sha Co. Ltd., Tokyo, Japan, 2007.
[4] A. Egyed. Instant consistency checking for the UML. In L. J. Osterweil, H. D. Rombach, and M. L. Soffa, editors, *ICSE*, pages 381–390. ACM, 2006.
[5] J. H. Hayes and A. Dekhtyar. Humans in the traceability loop: can't live with 'em, can't live without 'em. In *TEFSE '05: Proceedings of the 3rd international workshop on Traceability in emerging forms of software engineering*, pages 20–23, New York, NY, USA, 2005. ACM.
[6] A. Nöhrer and A. Egyed. Conflict Resolution Strategies during Product Configuration. In D. Benavides, D. Batory, and P. Grünbacher, editors, *VaMoS*, volume 37 of *ICB Research Report*, pages 107–114. Universität Duisburg-Essen, 2010.
[7] C. van Nimwegen, D. D. Burgos, H. van Oostendorp, and H. Schijf. The paradox of the assisted user: guidance can be counterproductive. In R. E. Grinter, T. Rodden, P. M. Aoki, E. Cutrell, R. Jeffries, and G. M. Olson, editors, *CHI*, pages 917–926. ACM, 2006.