User-Involved Preference Elicitation

Pearl Pu[1], Boi Faltings[2], Marc Torrens[2]
Swiss Federal Institute of Technology
[1]Human Computer Interaction Group
[2]Artificial Intelligence Laboratory
CH-1015 Lausanne, Switzerland

{pearl.pu, boi.faltings, marc.torrens}@epfl.ch

Abstract

When searching for multi-attribute services or products, understanding and representing user's preferences is a crucial task. However, many computer tools do not *afford* users to adequately focus on fundamental decision objectives, reveal hidden preferences, revise conflicting preferences, or explicitly reason about tradeoffs with competing decision goals. As a result, users often fail to find the best solution.

From building decision support systems for various application domains, we have observed some common areas of design pitfalls, which could lead to undesirable user behaviors and ineffective use of decision systems. By incorporating findings from behavior decision theory, we have identified and accumulated a set of principles for avoiding these design pitfalls: 1) provide a flexible order and choice in preference elicitation so that users can focus on fundamental objectives, 2) include appropriate information in a decision context to guide users in revealing hidden preferences, and 3) make tradeoff attributes explicit to facilitate preference revision and flexible decision making. We describe these principles and the corresponding interface *affordances*, and discuss concrete scenarios where they have been applied and tested.

1 Introduction

To perform complex tasks, such as searching the web for suitable products or services, planning a trip, or scheduling resources, people increasingly rely on computerized decision support systems to find outcomes that best satisfy their needs and preferences. However, automated decision support systems cannot effectively search the space of possible solutions without an accurate model of user's preferences. Preferences elicitation is therefore a fundamental problem of growing importance.

However, without an adequate interaction model and guidance, it is difficult for users to establish a complete and accurate model of their preferences. More specifically, we face the following difficulties:

- Inadequate elicitation tools can easily circumscribe a user in thinking about alternatives, thus forcing him to focus on means objectives rather than fundamental objectives. For

example, a user who commits to the choice of minivans (means objective) for spacious baggage space (fundamental) is not focusing on the values, and could risk missing alternatives offered by station wagons. In value-focus thinking, Keeney (1992) suggests that the specification and clarification of values should not be overtaken by the set of alternatives too rapidly.
- Users are not aware of all preferences until they see them violated. For example, a user does not think of stating a preference for intermediate airport until a solution proposes to change airplane in a place that he dislikes.
- Preferences can be ephemeral. For every trip, a user may state a different departure time preference.
- Preferences can be ill-defined. Users can state preference values that are potentially in conflict with values stated earlier.

There seems to be a dichotomy between what is required of a decision maker, that is being an expert of a domain and fluent with preference expressions, and the nature of his task, that is finding complex products in unfamiliar domains. The main problem in this dichotomy results from uncertainties in user's decision goals, and his lack of information on product characteristics. Therefore, the information flow from the system to the user must be unlocked in the preference elicitation process.

Other researchers both from behavioral decision theory and qualitative decision theory have also pointed out the advantage of eliciting user preferences incrementally. Mixed-initiative systems (MIS) are interactive problem solvers where human and machine intelligence are combined for their respective superiority (Allen *et al* 1994, Horviz 1999). MIS are therefore good candidates for incremental decision systems. Two main approaches have been reported in mixed-initiative systems literature: natural language dialogs or direct manipulation methods. An interaction model based on natural language dialogs will pose natural language questions to users and users in turn offer his input. The complex issue here is an adequate turn taking model. Furthermore, it is rather costly to construct dialog engines for each new domain.

We investigate a mixed initiative interaction model based on direct manipulation methods for decision support systems. The flexibility offered by this interaction model is key to effective user system collaboration. Each problem-solving agent (human or machine) contributes to the tasks that it does best, in the best opportune task context. More importantly, we found that the direct manipulation approach offers better user experience because it puts them in control (see also MacKay *et al* 2001).

Key to the direction manipulation model is a number of *affordances* that can engage and guide users for effective human/machine collaboration in preference elicitation. We are therefore preoccupied with the answer to the following questions:

1. Where to get information regarding decision making's behavior in order to understand the requirement of user interaction design?
2. What is the right order to elicit preferences for optimal user experience?
3. What information must be displayed to the user to stimulate preference expression (information design)?
4. When does ill-specified preference occur and how to guide users to revise them?
5. What are the most compact and cognitively sound display techniques to show attributes, preferences, and solutions to the user?

After developing a general framework and implementing four decision support systems, we have found some answers to the above-mentioned questions. We have accumulated a set of principles for designing the affordances, and have tested them with formative user studies throughout the implementation cycle.

This paper is organized as follows. We first address related works by describing the theoretical and conceptual foundation underlying our works, as well as comparing our works with similar ones in the area. Then we define the terms used in the modeling and solution generation formalism, since our decision problem is defined as an optimization search using constraint satisfaction techniques. We will define decision parameters from the user point of view before describing interaction methods for hidden preference elicitation, preferences revision and tradeoff analyses. We discussed several effective display techniques for solution space visualization, key to the decision context visualization used in a number of interaction principles. We then discussed our user studies, followed a section on our concluding remarks.

2 Related work

Qualitative decision theory aims at automating preference modeling and decision making processes, as well as extending traditional decision theories to cover larger contexts. Different approaches have been proposed (see Doyle and Thomason 1999 for a detailed survey), including the use of default and modal logic, and lately constraint satisfaction techniques (Boutillier *et al* 1997). Due to the uncertainty nature of user goals and preferences, incremental elicitation of preferences has received much attention, where a user's preference model is improved over time as he or she interacts with the system. Very little attention, however, has been paid to preference revision (Doyle and Thomason 1999). Most theoretical works have so far concentrated on the formalisms in which preferences are modeled, and the reasoning process underlying decision making. They have not been so far concerned with a decision maker's behavior and how to construct systems with effective user system interaction models.

Behavioral decision theory (Payne et al 1993, Carenini and Poole 2002), on the other hand, is very concerned with decision makers' behavior. Many years of studies have pointed out the adaptive and constructive nature of human decision making. The following key properties of a decision maker's behavior are fundamental validations to our hypotheses: 1) stating preferences is a process rather than a one-time enumeration of preferences that do not change over time; 2) user involved preference construction is likely to be more effective than using default or implicit models if a user is to understand and accept the solution outcomes (Carenini and Poole 2002); and finally 3) decision makers, when facing an unfamiliar task, are quite opportunistic in choosing their strategies by reassessing metagoals, and are likely to benefit from a flexibility to choose between fundamental and means objectives (see also Keeney 1992).

On the application side, several decision support systems have taken a similar approach, such as FindMe (Burke *et al* 1997), ATA (Linden et al 1997), and Apt Decision (Shearin and Lieberman 2001). All of them are concerned with decision support for the search of multi-attribute products. FindMe promotes assisted browsing relying knowledge of the domain, and aims at reducing complexities in the multitude of product dimensions and data sources for users. Tradeoff analyses and tweaking (vs. example critiquing) are two main components from their system that are similar to ours. However, we investigate a precise concept for the minimal critiquing context in which tradeoff can effectively happen, while their tradeoff component is mainly used for explanation, and tweaking for adjusting values. Apt Decision uses learning techniques to synthesize a user's preference model by observing their critique of apartment features. Its main objective is *profiling* and *predicting* what a user wants in apartment searches.

Linden et al (1997) described a preference elicitation method using travel planning as its example domain. Initially only few user preferences need to be expressed. The ATA system (automated travel assistant) uses a constraint solver to obtain several optimal solutions. Five of them are shown to the user, three optimal ones in addition to two extreme solutions (least expensive and shortest flying time). User preferences are modeled as soft constraints in the CSP formalism. A candidate critiquing agent (CCA), similar to our example critiquing, constantly observes user's modification to the expressed preference, and refines the elicited model in order to improve solution accuracy.

We have developed similar techniques for a range of applications rather than the travel domain alone, such as conceptual design, COMIND (Pu and Lalanne 1996, 2000, 2002), resource scheduling, ICARUS (Pu and Melissargos 1997), travel planning, SmartClient Travel (Pu & Faltings 2000, Torrens et al 2002), and vacation package finder, VacationPlanner (Jurca and Pu 2001). ATA displays only five solutions, which is not diverse enough to guarantee the inclusion of the most optimal solutions, especially when the current user deviates from the standard one (see Faltings, Torrens and Pu, the same issue). Furthermore, by not showing near optimal solutions, users are discouraged from opportunistic and creative discovery of outcomes. Using compact visualization methods, we are able to effectively display up to thirty solutions in SmartClient Travel, and more in other systems. Finally, we are more concerned with interaction *design principles* that can guide users to state hidden preferences, revise ill-defined ones, and focus on fundamental objectives. This is only possible by investigating user issues in a number of application domains.

Several recent works use constraint satisfaction techniques to automate the process of decision making. They also employ an interactive style to elicit preferences. Freuder et al (2000) introduced Exemplification CSPs by using an interactive version of the MAC algorithm to elicit user's value assignments to network parameters. O'Sullivan *et al* (2001) described machine learning techniques to help users formulate new constraints. The process involves asking users to provide acceptable solutions (positive examples) from which the system attempts to generalize the constraints exemplified. When the user finds a generalization not acceptable, then a negative example can be inferred to refine the version space of the constraint being acquired. Bowen (2001) provided a formal notion of interactive CSP, motivated by the introduction of additional constraints so that only a single solution is generated. A specialization CSP is a set of these constraints. We can see adding hidden constraints as a form of specialization CSP. We intend to incorporate these approaches and extend our category of principles. However, we must first validate them with user studies as we did with our own approaches.

3 Problem definition

We use constraint problem solving (CSP) techniques as an underlying reasoning engine for our decision support systems. This is largely due to the natural modeling of multiple attribute products and solution search within the CSP framework. A constraint satisfaction problem (CSP), which underlies the decision generation process, is characterized by a set of n variables $x_1,\dots,x_n$ that can take values in associated discrete domains $D_1,..,D_n$, and a set of m constraints $C_1,..,C_m$, which we assume to be either unary (one variable) or binary (two variables), and which define the tuples that are allowed for the variable or pair of variables. Besides integrity constraints that can never be violated, a CSP may also include *soft* constraints. These are functions that map any potential value assignment into a numerical value that indicates the preference that this value or value combination carries. Solving a constraint satisfaction problem means finding one, several or all combinations of complete value assignments such that all integrity constraints are satisfied. When soft constraints are present, it also involves finding optimally preferred assignments.

4 Decision parameters from user's point of view

We are concerned with a definition of attributes, criteria, preferences, and constraints from the user/system interaction point of view. A feature or component attribute is the aspect of a product (or solution) for which users would like to express preferences. A component attribute is a physical aspect of a product (e.g., the screen size of a PC), while a feature attribute is an abstract concept for a product such as the assemblability of a product (easy, hard). Optimization criteria are those attributes used for the evaluation of decision alternatives, and can be a function. For example, the total flying time of a trip is the sum of all of the flying segments plus the transfer time at intermediate airports. We can then state a criterion for the trip. For example, it should not take longer than x number of hours.

A preference is a statement about a desired condition on an attribute or a criterion. For example, "I would like to leave Geneva after 10am" is a user preference. Most user preferences are stated, although some can be inferred, such as those used in collaborative filtering. User's preferences can be also inferred from demographical analysis, default logic, and case-based reasoning (Ha and Haddawy 1998). For example, knowing that a traveler is booking a business trip, we can infer that his most important criterion is to be at destination on time. In this paper, we focus on user stated preferences only.

A constraint is a statement about a condition whose violation will lead to user's task failure. Being in Hamburg at 2pm for a business meeting is a constraint if that meeting cannot be pushed back. If the user fails to find a flight to satisfy that constraint, he will not be able to achieve his main objective. In most of our systems, user constraints are modeled as hard preference to achieve a uniform treatment of constraints and preferences. Besides user specified constraints, our system also deals with integrity constraints, those conditions that must be satisfied in the configuration process. For example, integrity constraints would state that there must be at least 30 minutes for changing flights within Europe, and 60 minutes for international ones. Integrity constraints are part of domain knowledge, and are elicited from domain experts during the construction of the system.

All of these variables are uniformly known as decision parameters, since they effect how users make decisions. We distinguish incomplete (or hidden) decision parameters from ill-specified ones. Ill-specified parameters give rise to conflicting solutions, assuming that attributes are not independent. Another ill-specified parameter can emerge when users are too concerned with means objectives rather than fundamental objectives (Keeney 1992). For example, a user immediately restricting himself to the choice of minivans will not be able to explore station wagon possibilities. If his fundamental objective is to have enough baggage space for his family, then the minivan preference is an ill-specified one.

5 Preference construction suitable for user's information needs

We surveyed 10 commercial on-line flight reservation systems (Jurca 2000) before while implementing our own travel planning system, SmartClient travel and subsequently Isy-travel (Pu and Faltings 2000, Torrens *et al* 2002). Almost all of the surveyed systems impose a fixed decision-making sequence on the user. As an example, consider Travelocity, a popular site for air travel. It requires a user to first fill in a form with the following information:

- Departure and destination airports
- Dates and times of travel
- Cabin class, and number of seats

- Preferences of airline companies

Users are required to state their preferences in the order reflecting the internal system architecture, rather than user's information needs of product information. For example, stating a desired departure time of 5pm means that none of the two daily nonstop flights will appear, as they both leave earlier in the day. On the other hand, stating a preference on the airline might mean that the only flights satisfying the airline constraint leave before 7 am, possibly too early for our traveler. Similarly, in hierarchically organized catalogs (e.g., www.pc-zone.com), buyers first have to answer a fixed sequence of questions corresponding to how databases are organized. This questionnaire is then sent to the seller's catalog server for product brokering.

The more serious problem is that a particular choice of attributes can easily force the user to state preferences using *means* rather than *fundamental* objectives. For example, suppose that a traveler lives near Geneva, and wants to be in Frankfurt by 3:00 pm (his fundamental objective). However, if he was asked to state departure time first, he would have to formulate a means objective. Unfamiliar with the available flights, he can only estimate the correct departure time, say 9 am. In this case, he would have missed the non-stop flight that leaves Geneva at 12 noon, which will get him there before 3:00 pm.

Furthermore, it is unclear what makes hidden preferences emerge for users in these existing systems. From our initial user studies, we noticed that users are able to critique example solutions, just as example solutions can provide system's knowledge to the user. We have therefore developed a technique, called example solution critiquing, that enables users to *reveal* hidden preferences while providing decision context at the same time. Further, an example solution gives users a direct contextual mapping between preferences and consequences, thus providing more knowledge of the consequences of their preference statements.

Since example critiquing has become a central component in our preference elicitation process, we are also concerned with example solutions that are too complex to display and too puzzling for users. We thus define the notion of minimal context for which critiquing could effectively happen.

5.1 Any preference, any order elicitation

In the most obvious case of stating preferences, a user applies preferences by selecting value assignments or restricting domains of individual variables. The CSP engine is used as a consistency maintenance mechanism, and generates feasible decision alternatives. Each preference can be a decision variable.

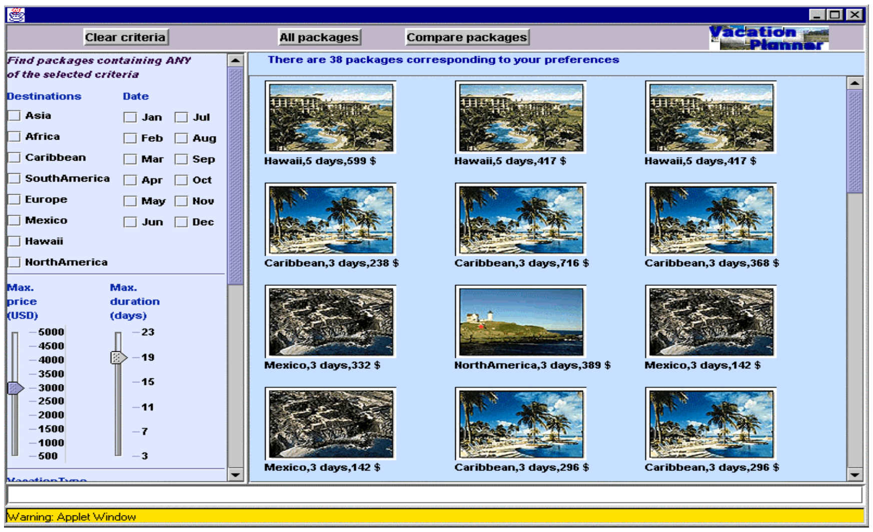


Figure 1: Users specify attribute preferences in any order, at any time in the elicitation process

Figure 1 is an interface of VacationPlanner using our framework (Jurca and Pu 2001). All attributes of a vacation package (destination, date, max price, max duration, youngest accepted age, and etc.) are displayed on the left-side panel. Users express preferences on these attributes by clicking on the desired check boxes or moving sliders into the right positions. The matched vacation packages are generated and displayed instantly on the right-side panel.

While testing the VacationPlanner, we observed that because users are able to choose the preferences that correspond to their main objectives, and state values whose consequences are immediately shown, they can more effectively and quickly establish an accurate preference model. There is a strong contrast between systems that we have surveyed such as ClubMed.com, and VacationPlanner. For example, a user, looking for a ClubMed village to spend two weeks with his children, was not able to indicate that he has a child younger than 2 years old. Only after he has chosen a village, did he realize that small children are not allowed.

We have identified and tested the following three interaction design principles:

1. Any preference, any order: To enable users to focus on fundamental objectives, the key is to allow them to choose any preference in any order. The incremental elicitation process should not follow the structure imposed by the system architecture.
2. Immediate feedback: Since some preference values do not have an obvious relationship with decision outcomes, an immediate feedback of results is important and allows users to correctly access how to proceed with the incremental process.
3. Visual feedback: Since users have to quickly see and evaluate the outcome in the preference -> outcome loop, a visual feedback is more effective than a textual list of solutions (for both vacation and trips) because seeing is more preferred than reading.

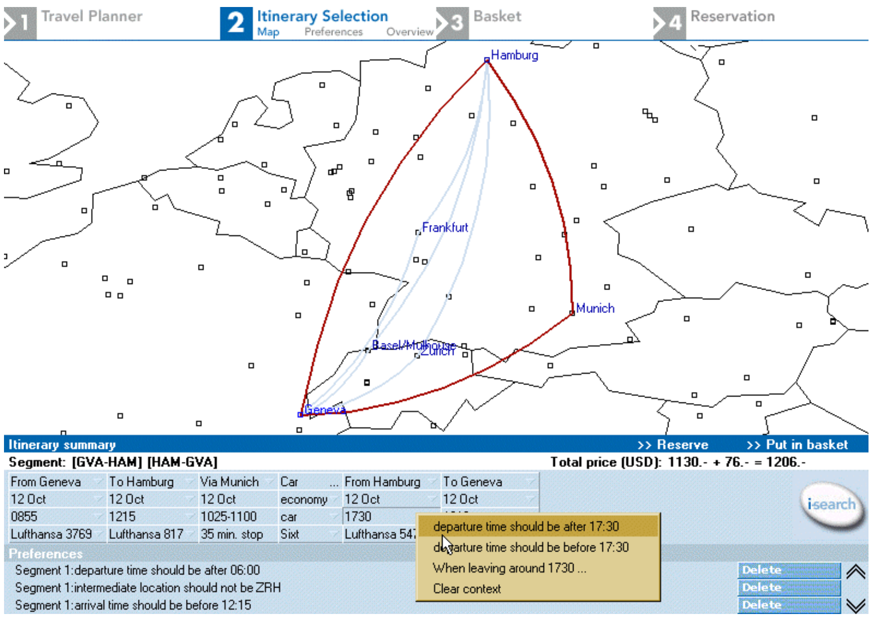
5.2 Preferences via example critiquing

In SmartClient- and Isy-Travel, a user only has to state the destination city (not airport), and a range of dates as initial data. A search engine then fetches flight information from a central database (a global distribution system GDS), and calculates a total of 30 solutions. Only one of them is displayed in detail, although others are easily accessible without contacting the GDS again.

Figure 2 shows an example solution for a trip between Geneva and Hamburg. Here the user clicks on the departure time attribute for the return trip, since his meeting is likely to last beyond 17:30. In the subsequence interaction, he also expresses a preference for the departure time, since he would like to be at Hamburg between 11am and 6pm. As he critiques on attribute values, the system incrementally refines the solution quality.

We found that most users were unable to state other preferences beyond the very fundamental ones. On the other hand, it is easier for them to critique examples, especially those showing violations. Thus, a system that simply asks users to state their preferences cannot effectively obtain a complete list. We call an example solution in which one or several preferences have been violated a critiquing context (CC). We have the following principles for example critiquing:

1. <u>React to examples:</u> Most users are not aware of or cannot articulate preferences. However, they can critique examples illustrating preference violations, and become aware of them. The hidden preference is most effectively solicited via example solutions and users' direct critique on the attributes.
2. <u>Show decision context:</u> Most users like to see a larger context than a localized one in which he would state preferences. The most effective decision context is an example solution in which attributes can be directly manipulated, and the decision consequences (effects) immediately calculated.



Segment: [GVA-HAM] [HAM-GVA]

Total price (USD): 1130.- + 76.- = 1206.-

From Geneva	To Hamburg	Via Munich	Car ...	From Hamburg	To Geneva
12 Oct	12 Oct	12 Oct	economy	12 Oct	12 Oct
0855	1215	1025-1100	car	1730	[illegible]
Lufthansa 3769	Lufthansa 817	35 min. stop	Sixt	Lufthansa 547	

Preferences

Segment 1:departure time should be after 06:00

Segment 1:intermediate location should not be ZRH

Segment 1:arrival time should be before 12:15

Figure 2: A critiquing context is a solution that violates user's preferences

5.3 Minimal critiquing context

In most cases, it is effective to use an entire example solution as a critiquing context, because it accommodates all attributes and it is easy to implement. There are two situations for which showing an entire solution is not ideal: 1) when a solution contains too much detail to show, and

2) attributes influencing decision objectives are not explicit in the example solution. For the first situation, business trips for example may well contain more than the usual outbound and inbound segments. Flying from Geneva to San Francisco with a visit to Denver for a couple of days is a three-segment trip. In the second situation, choosing a PC feature, such as advanced rather than (rudimentary, basic, high volume), entails to a certain configuration of PC components. When displaying a solution to users, the feature attributes do not become explicit in the example solutions.

We define the dependent attribute cluster to be a set of dependent attributes whose values influence each other in the decision process. Such a cluster is the minimal critiquing context. If a decision problem entails a number of such clusters, we should allow users to critique them one at a time for an effective management of display complexity and cognitive load. As an example, we can have several critiquing contexts in the travel domain:

- cabin class, price
- departure time, arrival time, airline, intermediate airport for the outbound
- departure time, arrival time, airline, intermediate airport for the inbound

This is because the choice of cabin class influences price and vise versa. Departure time, on the other hand, influences the arrival time, the choice of airlines, and the intermediate airport (if they are available). Given a CSP based decision system, it is possible to detect such clusters either by examining the network structure, inferring these cluster structure from solutions (not trivial), or performing domain knowledge engineering.

In practice however, attributes tend to be related in a complex dependency structure. In the above example, airline can be an attribute in the first group as well if we assume that airlines offer very different fares. If they are strictly modeled as dependent, then the three clusters will join and the minimal critiquing context becomes again the entire solution, thus possibly too complex for the multi-segment trips. A possible solution is to perform conditional preference elicitation. That is, after a user has stated a preference on airline, then other attributes would become independent and form clusters (see conditional preferences in Boutilier et al 1997). However, this forces users to state preferences in a particular order.

3 Show minimal context: if attributes form smaller dependent clusters, it is more optimal to use each cluster as the minimal critiquing context. This reduces the cognitive and display complexities for users.
4 Show feature attributes in critiquing context: if preferences were elicited on feature attributes, then the critiquing context should be enlarged to accommodate them as well.

6 Preference revision and tradeoff analysis

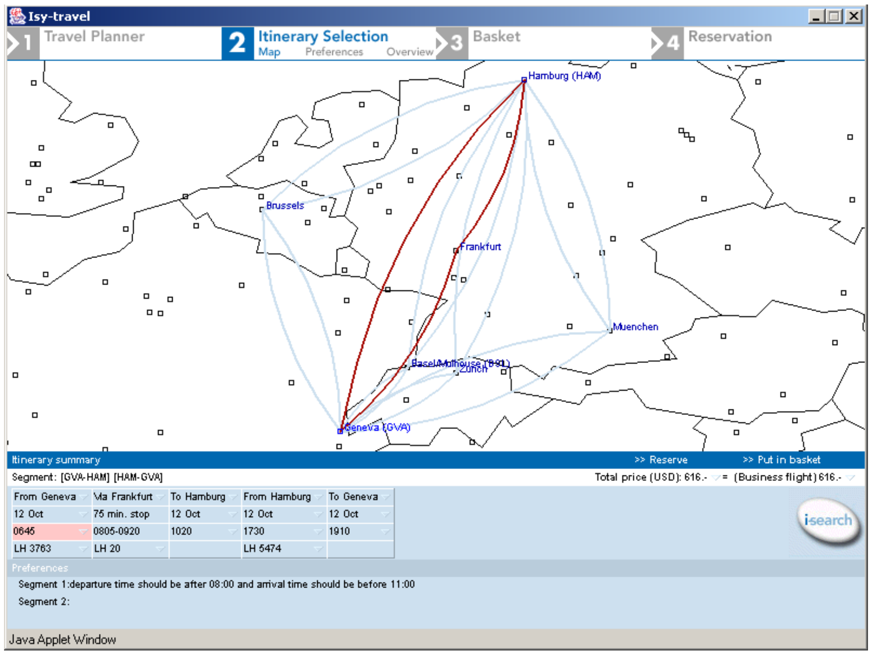
A previous stated preference should be modifiable in the elicitation process. We identify two cases where revision is necessary: stated preferences are not accurate enough for the user model; and a stated preference gives undesirable consequence to another dependent attribute. The first case reflects cautious users who state initially modest preferences and constraints. This gives rise to a Pareto optimal set that is too large to browse. Further refinement is necessary. The user in this case continues in preference elicitation process. The second case is more difficult where conflicting values of competing attributes must be revised and tradeoff analysis carried out in order to obtain desirable solutions (Keeney and Raiffa 1976).

Besides following the (minimal) critiquing context principle to offer an explicit explanation of the competing attributes, there are two ways to perform tradeoffs at the attribute level, or at the structure level.

6.1 Attribute level tradeoffs

In SmartClient- and Isy-Travel, users are allowed to pose inconsistent preferences. CSP algorithms for partial satisfaction are used to compute near-best solutions where preferences over certain attributes have been violated. An explanation mechanism is used to highlight dependent attributes and their respective assigned values. Users can choose to keep the proposed solution, or retract the inconsistent preferences.

In Figure 3, a user has stated a preference on both departure and arrival time for segment 1. He usually prefers to fly out of Geneva after 8:00 am, but for this trip he has to be at Hamburg before 11 am. The flight shown violates the first preference, but satisfies the second. The violated attributed has been highlighted. At this point, the user must decide whether he will leave earlier, or he can push back the meeting time. In the parallel coordinate display (Figure 7), it is easier to see that the flight shown is the best solution. We discuss this display technique in a later section.



Segment: [GVA-HAM] [HAM-GVA]

Total price (USD): 616.- = (Business flight) 616.-

From Geneva	Via Frankfurt	To Hamburg	From Hamburg	To Geneva
12 Oct	75 min. stop	12 Oct	12 Oct	12 Oct
0645	0805-0920	1020	1730	1910
LH 3763	LH 20		LH 5474	

Preferences

Segment 1:departure time should be after 08:00 and arrival time should be before 11:00

Segment 2:

Figure 3: Conflict values can be revised via tradeoff analysis

Alternatively, users can use the attribute editing feature similar to the one used in VacationPlanner (left panel in Figure 1) to perform tradeoff. When a user states a low-budget preference on price, for example, he sees immediately that vacation packages for Hawaii disappears. At this point, if he decides that going to Hawaii is a more fundamental objective, he can revise the price preference until he is satisfied with the set of alternatives.

6.2 Structural level tradeoffs

The above design principle suits an average user who does not intent to know the underlying constraint network and how that structure has caused conflicts. In a configuration design system that we built, COMIND (Pu and Lalanne 1996, 2000, 2002), one of the major user's tasks is to establish a structural model of attributes, and their adequate dependencies. In this case, preference revision involves revising the underlying constraint structure, as well as making tradeoffs among attribute values.

In COMIND, preferences that are in conflict are explained in a lattice, shown as the stacked tiles in Figure 4. Each tile represents a set of preferences. A black tile indicates that preferences represented by that set cannot be satisfied simultaneously, thus called the conflict set. Shades of blue are used to indicate the constrainedness of a set. The darker it is, the preferences are more strongly constraining to each other; that is, yielding few admissible solutions. Consider the map-coloring example with three countries and the same two colors for each country. If two attributes are considered at a time, there are no conflicts. When three sets of attributes are considered, a conflict is detected. Clicking on that square, the system shows the set in detail, and if the subsets (shown as white squares in the lower lattice). COMIND offers a designer to choose smaller sets to diagnose (easier but less rewarding) or the conflict set itself. In the case of the map-coloring problem, revising two-attribute sets is not very effective since there are no trivial conflicts. Revising the three-attribute set allows users to get to the bottom of the problem. That is, more values need to be added for at least one of the attributes. Finally, there is an option to edit the structure in the network editor (Figure 5), which is not possible in the map-color example, but feasible in conceptual design. For more details on conflict resolution, see Lalanne (1998).

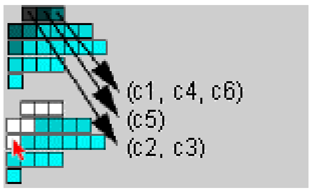


Figure 4: A lattice with conflicting sets (black), constrained sets (dark blue)

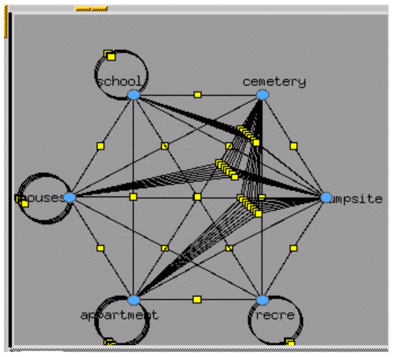


Figure 5: Constraint network structure visualization

We therefore propose the following principles to resolve conflicts at the network structure level:

3 Critiquing and minimal critiquing contexts– these two previous principles are still valid interaction mean for the visualization of conflicts, thus enabling users to revise preferences at the attribute level.
4 Tradeoff at the structure level: when users have to resolve over-constrained problems, doing this at the level of network structure is more effective than at the level of attribute values in potentially a huge number of individual solutions. That is because structures affect all solutions uniformly.

7 Solution space visualization

Finally, displaying effectively the solution space is crucial, especially for showing users the consequences on stated preferences, and the decision context for eliciting preferences. The solution space can be enormous for multi-attribute products due to the combinatorial nature of how attributes can be combined. In considering the interaction design principle, we face the tradeoff between displaying a small number of solutions and too many. The former would risk not displaying the optimal solutions, while the later approach would overload users visually and cognitively. See Faltings *et al*, same issue, for some conclusions regarding the most optimal number of solutions to display. We derived two important requirements to address these concerns: compactness and organization. Compactness is useful for displaying many solutions, while organization allows users to zoom into a subspace easily. In each of the solution visualization discussed earlier, the organization was achieved by natural groupings found in geographical properties of solution spaces, such as the geographical regions for vacation packages, and flight routes for travel planning. Here we introduce a space-filling overview method that displays any types of solutions in hierarchical structures.

Figure 6 is a TreeMap (Johnson and Shneiderman 1991) visualizing the entire solution space by recursively cutting a square into subparts. There are four squares after the first vertical cut with three longest and heaviest vertical lines. Each subsequence cut (alternate the direction of cuts) results in smaller squares. This recursive way of cutting the space results in a hierarchical structure. We experimented with the use of the ID3 algorithm to structure the solution space before applying the TreeMap technique. Each cut of the solution space corresponds to a preference value. The smallest square corresponds to a decision outcome whose characteristics have been described the location of that tile in the space. The tile in yellow, for example, is a reassignment of an aircraft in a particular rotation, PL66. This method is especially effective when the solution space is big, and users can only express preferences in the solution space.

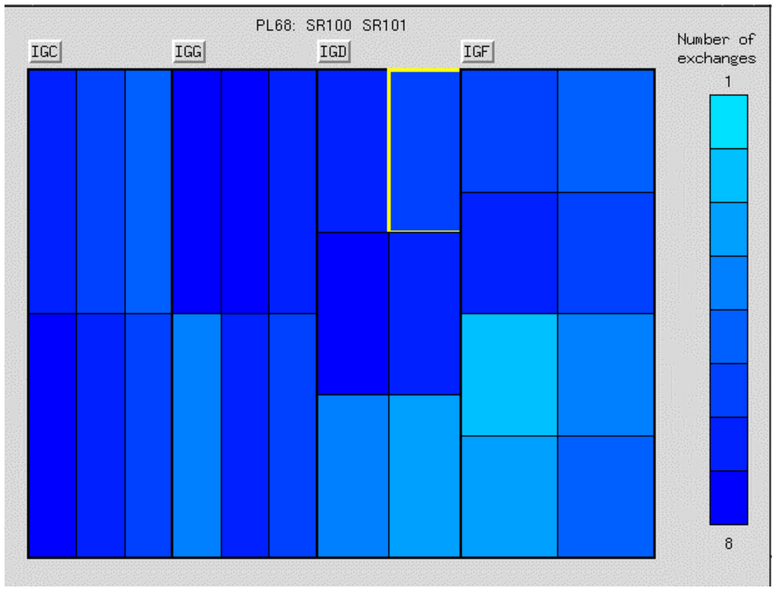
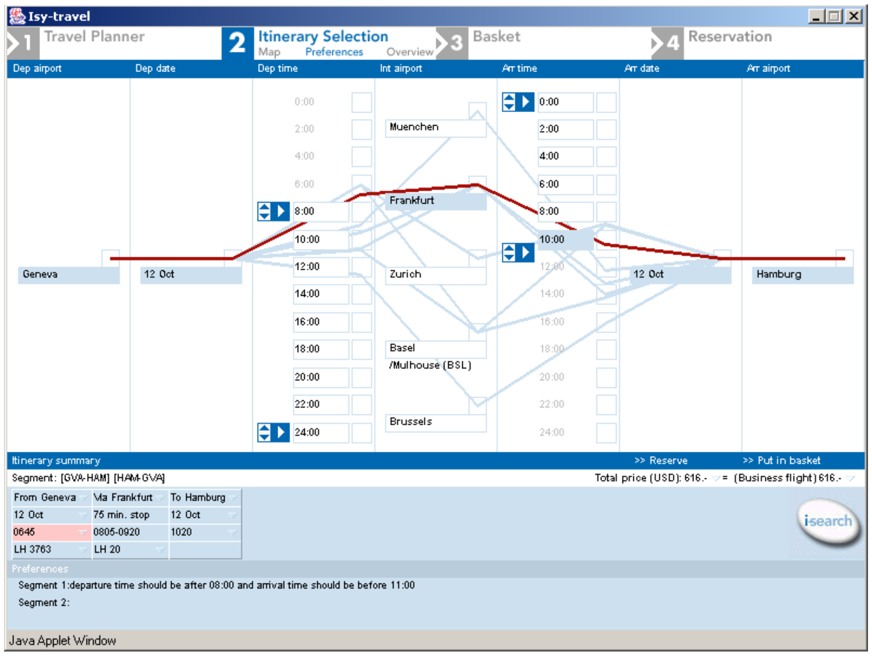


Figure 6: Treemap is used to indicate user preference for aircraft rescheduling

Another compact visualization is called parallel coordinates (Inselberg and Dimsdale 1990). Figure 7 is the implementation of that method for the travel domain. All attributes within the minimal context are represented simultaneously. A preference on one attribute will cause the adjustment of values on all of the other attributes. For example, currently the user has indicated a range of arrival times. The values for departure and intermediate airports are calculated and displayed. The departure and arrival airports are not part of the minimal context, but are there to make the context more readable. Lines tracing through these attributes show possible solutions. This method is very effective for the manipulation of attribute values, and visualizing a concrete example solution in the same display.

A potential disadvantage of this display is the mixture of discrete attributes (nominal variables) with ordinal and numerical ones. The intermediate airport, a nominal attribute, disturbs the visual semantics of the other attributes. That is, departure and arrival times are numerical and ordinal variables. Two points in the display indicates an order (in this case, top to bottom means earlier to latter in time). But airports cannot be ordered that way. To fix that problem, we dynamically display airports so that admissible ones are grouped continuously together, while the ordering reflects certain optimality such as shortest flying time. There are other solutions that address the problem of mixture of nominal and ordinal variables.



Segment: [GVA-HAM] [HAM-GVA]

Total price (USD): 616.- = (Business flight) 616.-

From Geneva	Via Frankfurt	To Hamburg
12 Oct	75 min. stop	12 Oct
0645	0805-0920	1020
LH 3763	LH 20	

Preferences

Segment 1:departure time should be after 08:00 and arrival time should be before 11:00

Segment 2:

Figure 7: attribute ranges and solution space in one display

8 User studies: combining decision behavior research with user requirement analysis, and formative usability studies

Some existing findings from consumer behavior research and behavioral decision theory allowed us to avoid costly experiments to understand users' behaviors when they face decision tasks. However, we still analyzed usability issues by surveying existing commercial systems, such as the survey (Jurca 2000). We also used online decision systems such as Personalogic in our user requirement studies. The characteristics, both in terms of shortcomings and strengths, were integrated into the design of Isy-travel and VacationPlanner. Earlier decision support systems were built from analysis done on commercial resource allocation systems employed in the airline industry. COMIND was based on extensive interviews with conceptual designers. The design principles reported here are therefore a result of consumer behavior literature, user requirement analyses, and most importantly testing of these principles in systems that we have built.

Formative user studies were also performed for all of the systems described here. It's a process involving writing scripts for each usage scenario, identifying potential users, conducting usability testing, analyzing results, and proposing design changes. The systems all went through several phases of change as a result of formative user studies.

For the future, we intend to conduct several comparative user studies. For example, we want to compare the nature of elicitation under the condition where users actively volunteer preference via critiquing, and under the condition where they are prompted by natural language questions to express preferences. Besides investigating the effectiveness of active and passive preference stating, we would like to also compare how helpful the two modes are in terms of finding "best" solutions.

9 Conclusion

A number of researchers from both behavioral and qualitative decision theory have pointed out the advantage of eliciting user preferences incrementally. We have shown the effectiveness of a general interaction model based on mixed initiative approaches for building such incremental decision support systems. A set of design principles have been described, which aim at *affording* and *enabling* users to state hidden preferences, revising conflicting preference values, and flexibly choose between fundamental and means objectives. These principles have been accumulated, clarified, and tested from building four interactive decision support systems. Further, we found that our direct manipulation and visualization approach offer better user experiences because it puts them in control while providing them with knowledge from data sources at the same time. Even though we have covered many relevant scenarios, we do not claim that the categories are exhaustive. This is a first step towards building the set of principles, which should be extended further to eventually provide a library of GUI designs for decision and related systems.

10 Reference

J. F. Allen, L. K. Schubert, G. M. Ferguson, P. A. Heeman, C. H. Hwang, T. Kato, M. N. Light, N. G. Martin, B. W. Miller, M. Poesio, and D. R. Traum. *The TRAINS project: A case study in building a conversational planning agent*. TRAINS Technical Note 94-3, University of Rochester, Department of Computer Science, September 1994.

C. Boutilier, R. Brafman, C. Geib, and D. Poole. *A Constraint-Based Approach to Preference Elicitation and Decision Making*. In AAAI Spring Symposium on Qualitative Decision Theory, Stanford, 1997.

Bowen, J. *The (Minimal) Specialization CSP: A basis for Generalized Interactive Constraint Processing*, in Proceedings of Workshop on User-Interaction in Constraint Processing at CP-2001. 2001.

R. Burke, K. Hammond, and B. Young. *The findme approach to assisted browsing*. In IEEE Expert, volume 12(4), pages 32--40, 1997.

Carenini G. and Poole D. *Constructed Preferences and Value-focused Thinking: Implications for AI research on Preference Elicitation*. **AAAI-02 Workshop** on Preferences in AI and CP: symbolic approaches - Edmonton, Canada, 2002.

J. Doyle and R. Thomason. *Background to qualitative decision theory*. AI magazine, 20(2), summer 1999.

Freuder E, Likitvivatanovong C and Wallace R. *A case Study in Explanation and Implication*, in Proceedings of CP-2000 Workshop on Analysis and Visualization of Constraint Programs and Solvers, 2000.

Jurca, A. and Pu, P. *Algorithms for Online Vacation Planning*. Technical Report, Swiss Federal Institute of Technology, Lausanne. p. 1-12, 2001.

Jurca, A. *Survey of Online Travel Planning Systems.* Technical Report, Swiss Federal Institute of Technology, Lausanne, 2000.

Keeney, R. and Raiffa, H., *Decision with multiple objectives: Preferences and value tradeoffs*. 1976: Cambridge University Press.

Keeney, R., *Value-Focused Thinking: A Path to Creative Decision Making*. 1992: Harvard University Press.

V. Ha and P. Haddawy. *Problem-focused incremental elicitation of multiattribute utility models*, in Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence, pages 215--222, August 1997.

Ha, V. and Haddawy, P. *Toward Case-Based Preference Elicitation: Similarity Measures on Preference Structures*, in Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence, 1998.

Horvitz, E., *Principles of mixed-initiative user interfaces*, in Proceedings of The ACM SIGCHI Conference on Human Factors in Computing Systems, ACM Press, p. 159-166, 1999.

Inselberg, A. and Dimsdale, B. *Parallel Coordinates: a Tool for Visualizing Multidimensional Geometry*, in Proceedings of IEEE Visualization '90, IEEE Computer Society, p. 361-378, 1990.

Lalanne, D. *Computer Aided Creativity and Multi-criteria optimization in Design*. Ph.D. thesis, the Swiss Institute of Technology (EPFL), Lausanne, 1998.

G. Linden, S. Hanks, and N. Lesh. *Interactive assessment of user preference models: The automated travel assistant*. In Proceedings of User Modeling '97, 1997.

Mackay, W.E., Holm, E., and Horn, S.B., *Who's in Control? Exploring human-agent interaction in the McPie Interactive Theater project*, in Proceedings of The ACM SIGCHI Conference on Human Factors in Computing Systems. 2001.

Payne, J.W., Bettman, J.R., and Johnson, E.J., *The Adaptive Decision Maker*. Cambridge University Press, 1993.

Pu, P. and Faltings, B. *Personalized Navigation of Heterogeneous Product Spaces using SmartClient*, in Proceedings of the International Conference on Intelligent User Interfaces, ACM Press, 2002.

Pu, P. and Faltings, B. *Enriching Buyers' experiences: the SmartClient Approach*, in Proceedings of The ACM SIGCHI Conference on Human Factors in Computing Systems, 2000.

Pu, P. and Lalanne, D. *Human and Machine Collaboration in Creative Design*, in Proceedings of the European Conference of Artificial Intelligence, 1996.

Pu, P. and Lalanne, D. *Interactive Problem Solving via Algorithm Visualization*, in Proceedings of the IEEE Information Visualization Symposium, IEEE Press, 2000.

Pu, P. and Lalanne, D. *Design Visual Thinking Tools for Mixed Initiative Systems*, in Proceedings of the International Conference on Intelligent User Interfaces, ACM Press, 2002.

Pu, P. and Melissargos, G. *Visualizing Resource Allocation Tasks*. IEEE Computer Graphics and Applications, 1997. **17**(4).

O'Sullivan, B., Freuder, E., and O'Connell, S. *Interactive Constraint Acquisition*, in Proceedings of Workshop on User-Interaction in Constraint Processing at the CP-2001, 2001.

Shearin, S. and Lieberman, H. *Intelligent Profiling by Example*. in Proceedings of the Conference on Intelligent User Interfaces, ACM Press, 2001.

Johnson, B. and Shneiderman, B. *Tree-Maps: A Space-Filling Approach to the Visualization of Hierarchical Information Structures*, in *Visualization '91*, p. 284-291, 1991.

Torrens, M., Faltings, B., and Pu, P., SmartClients: Constraint Satisfaction as a Paradigm for Scaleable Intelligent Information Systems. International Journal of Constraints, 2002. **7**: p. 49-69.