Part III

User Explanation

As shown in the previous part, our goal is to provide a high-level way for users to specify their preferences so that an agent, provided with the ability to reason about these preferences and making decisions, can act on their behalf or support their decisions. However, users are willing to delegate their tasks to personal agents, only if: (i) they know exactly what the agent is going to do (Schiaffino and Amandi 2004); or (ii) users have built trust in the agent based on previous interactions with it. However, there is still lack of an in-depth investigation of the concrete system design features that could be developed to promote user trust (Chen and Pu 2010).

One of the most investigated ways of increasing user trust in their personal agents, which may be an expert or recommender system, is providing agents with the ability of giving explanation for users to justify their decisions. When an autonomous agent makes decisions based on user preferences, which may be complex, conflicting, and changing, it can be obscure and difficult for the user to understand and know why the agent has made certain decisions. Without understanding whether there is a rationale behind an action, the user is unlikely to trust the agent to act on her behalf, and consequently accept the system.

We begin this part by presenting a literature review (Chapter 7) of different approaches that aim at providing explanations for users about decision making and recommendation methods in order to increase user trust in decisions and recommendations. Then we describe a study (Chapter 8) performed to identify the kinds of arguments that people make to support a decision. This study allowed us to derive explanation templates and guidelines to be addressed by decision making methods and recommender systems. Finally, we present an approach (Chapter 9) for generating explanations, which follows the proposed templates and guidelines, and which is based on execution traces produced by our decision making technique described in the previous part.

7
Background on Explanation Approaches

The need for explaining why systems perform in a certain way or make particular decisions has emerged with the popularity of Expert Systems (ESs) in the 80's. More recently, explanations have been explored in the context of other areas that involve decision making, such as recommender systems. In this chapter we review different approaches proposed for justifying decisions made by computer systems by means of explanations. First, we overview how explanations were addressed in the context of ESs (Section 7.1). Next, we discuss explanation in recommender systems (Section 7.2), which are considered the successors of ESs (Tintarev and Masthoff 2007). Then, we describe approaches that focus on explaining relaxations performed in over-constrained problems (Section 7.3), followed by general approaches for explanations of decision making techniques based on multi-attribute preference models (Section 7.4). Finally, we conclude in Section 7.5.

7.1
Expert Systems: the Roots of Explanation

Expert Systems (ESs) are computer programs whose aim is to automate human decisions of a particular application area, and incorporate expert knowledge of this area. An ES consists of (a) a reasoning engine, (b) knowledge about the domain (which is typically captured in the form of rules), and (c) a knowledge base, which together are able to make decisions in a similar way that humans do, such as medical diagnostics. With the popularity of such systems, researchers and software developers observed that systems able to make decisions without giving an explanation of how an answer is obtained are very unlikely to be accepted, as users want to understand the rationale behind the choice to trust in it, and possibly to be able to justify the decision for other individuals. This intuition of the need for explanations is confirmed by the empirical study performed by Ye and Johnson (Ye and Johnson 1995), which concluded that explanation facilities indeed have a positive impact on the user acceptance of an ES's advice.

Different ways of providing explanations have been proposed, which were classified by Nakatsu (Nakatsu 2006) into three categories: (i) reasoning traces;

(ii) strategic knowledge; and (iii) deep justifications. The first (*traces*) is the most straightforward to be obtained. Most of the existing ESs are rule-based systems, and therefore the reasoning consists of applying rules according to a specific input in order to obtain a result. Each rule application can be seen as a trace of the reasoning process, which can be used to explain the decision made. These traces can be used to answer different types of questions made by user, such as "*why*" (why the system is asking a particular kind of information), "*how*" (how the conclusion was obtained), "*why not*" (why the system did not reach a particular conclusion). The first two types of questions are widely known, as they were addressed by one of the most famous ESs, namely MYCIN (Buchanan and Shortliffe 1984). More sophisticated forms of this approach consist of transforming computer-generated code to natural language, so that users can better understand displayed rules, and also omitting details according to the user knowledge.

The second type of explanation, *strategic* knowledge, consists of making the problem-solving strategies explicit to the end user. Therefore, the explanation is less detailed than with the use of reasoning traces, and solely the overall line-of-reasoning is made visible to the end user. What is presented for the users is a strategic knowledge structured for the end user, which can be a tree of goals, or a tree of topics. The third explanation type, deep *justifications* (also known as "*canned text*"), is an explicit description of the causal argument or rationale behind each inferential step taken by the ES. In this case, an explanation is tied to an underlying domain model that provides structural knowledge and taxonomic knowledge about the domain.

Besides showing the impact of explanations on the user acceptance of ESs, the study of Ye and Johnson (Ye and Johnson 1995) also investigated the impact of these three different types of explanation, and concluded that the most effective one is *justifications*. Nevertheless, it has the drawback of requiring developers to predict questions that users are going to make, and also codify answers, in order to produce human-like explanations.

Many ESs were implemented as a proof of concept of many of the ideas related to ES explanations, and they are implementations of these three kinds of explanations introduced above. For further information about implemented ESs that support explanation, we refer the reader to the review made by Lacave and Diez (Lacave and Diez 2004).

In addition, there are works that focus on other aspects of the incorporation of explanations into ESs, such as the kinds of arguments to be presented to users. Bohanec and Rajkovič (Bohanec and Rajkovič 1993) have proposed a different form of reasoning for ESs, which integrates ideas of multi-attribute decision making. They define the knowledge representation with a tree of criteria. Leafs of the tree are referred to as *basic* criteria, and the remaining ones are called *aggregate* criteria. For example, as illustrated in the paper, an aggregate criterion, which is part of the criteria tree for the evaluation of microcomputer hardware, is "Basic Communication," which is composed of two basic criteria "Keyboard," and "Monitor." Each of these criteria is associated with an evaluation domain, such as $\{bad, acceptable, good, excellent\}$, and aggregate criteria are specified in terms of the values given for basic criteria, for instance, if a computer has a *good* monitor and an *unaccept* keyboard, the basic communication is specified as *unaccept*. With these "preferences" specified, one can make a decision to chose the "best" option, and give an explanation, based of the criteria that makes one option better than another. This approach has the limitation of not addressing how this information is obtained — as the authors state, they left the *knowledge acquisition* out of the scope. Nevertheless, their approach requires the specification of how each basic and aggregate criterion is evaluated, and this involves a combinatorial problem, thus requiring this information from users is infeasible.

On the other hand, some works have given attention to the architecture of ESs integrated with explanation ability. Shankar and Musen (Shankar and Musen 1999) have proposed a multi-agent framework, named WOZ, that provides explanations by using arguments. The framework proposes the use of a set of components, each having a particular responsibility in the explanation process, such as a user model that specifies the characteristics of the current user (e.g. the user's protocol expertise, domain expertise, familiarity with patient data, and visual preferences), and the explanation strategy, which contains the situation-action mappings. A "director" component is responsible for orchestrating all the other components. The framework proposes to make the explanation visualisation independent from its generation, but it is abstract in that *it is does not specify how the explanation is generated*.

The approach consists of having a base of *meta-arguments*, which conceptualise arguments for a *class of claims*. The elements of the meta-argument structure are stated with abstract descriptions. These meta-arguments are used to produce *concrete arguments* (instances of meta-arguments), which define arguments for a specific claim of a class of claims. Meta-arguments, and consequently arguments, follow a particular structure — the Toulmin's Argument Structure (Toulmin 2003) — whose components are detailed in Table 7.1.

El-Beltagy et al. (El-Beltagy et al. 1999) focused on an architecture based on agents, which separates different concerns related to explanation. The agents related to the explanation generation are: (i) terminology server, which provides terminology definitions and abstractions, as well as detailed term knowledge within a given predefined domain; (ii) "why" agent, which is responsible for justifying

Table 7.1: Toulmin's Argument Structure.

Component	Definition
Data	the particular facts about a situation on which a claim is made
Warrant	the knowledge that justifies a claim made using the data
Backing	the general body of information or experience that validate the warrant
Qualifier	the phrase that shows the confidence with which the claim is supported to be true
Rebuttal	the anomaly that shows the claim not to be true
Claim	the assertion or conclusion put forward for general acceptance

the question asking strategies using familiar domain-oriented knowledge; and (iii) domain-oriented justification agent, which generates a "plausible story" from the given inputs that justify the conclusion. The proposed approach does not detail how the responsibilities given for each agent are performed, and therefore the explanation generation itself is not described.

Said et al. (Said et al. 2009) argued that the form adopted to provide reasoning for ESs (or problem solving method) is tightly coupled to how explanations are generated, so they proposed a methodology for automatically generating explanations during and at the end of the reasoning process, considering different knowledge representation schemes, which are "generate and confirm hypotheses," and the "routine design generic task." They show, in an *abstract way*, which kind of information should be displayed as an explanation for each of these two approaches, for two different explanation primitives: (i) "why asking," in which users want to get an explanation for why the system is asking a certain question; and (ii) "how," so at the end of the consultation the user may want to get an explanation of how the system reached a certain decision or conclusion.

As we mentioned before, these works focused on general approaches for explanation (not on the implementation of particular systems) and architectures; however, they are abstract in a way that the explanation generation still remains as a challenge.

7.2
Explanation in Recommender Systems

Some researchers have referred to *recommender systems* as the successors of ESs (Tintarev and Masthoff 2007). Since their beginning, approaches for recommender systems have aimed at increasing the accuracy of such systems, in order to evaluate their quality. More recently, researchers have perceived the relevance of explanations, as it was the case with ESs. Tintarev and Masthoff (Tintarev and Masthoff 2007, Tintarev and Masthoff 2011) have performed a survey on explanations in recommender systems and have identified seven different aims related to approaches proposed for explanation, which are shown in Table 7.2.

Recommender system approaches are classified into three categories,

Table 7.2: Aims of Recommender Systems (Tintarev and Masthoff 2007).

Aim	**Definition**
Transparency	Explain how the system works
Scrutability	Allow users to tell the system it is wrong
Trust	Increase users' confidence in the system
Effectiveness	Help users make good decisions
Persuasiveness	Convince users to try or buy
Efficiency	Help users make decisions faster
Satisfaction	Increase the ease of usability or enjoyment

which are associated with the technique used to produce recommendations (Adomavicius and Tuzhilin 2005): (i) *content-based approaches*, which recommend to users products that are similar to the ones the user preferred in the past; (ii) *collaborative approaches*, which recommend items that people with similar tastes and preferences liked in the past; (iii) *hybrid approaches*, which combine collaborative and content-based methods. In addition, some approaches are said *preference-based*, as they aim at predicting an implicit ordering among products instead of predicting a rate that the user would give to the product.

Explanation approaches that are proposed for recommender systems address the different presented aims by providing explanations that show the rationale behind the adopted recommendation approach. For instance, if a collaborative approach is adopted, the user may receive as an explanation a histogram of ratings of the product given by similar users. In addition, simple clarifications, such as "*people who bought this product also bought X*," are considered effective.

There are other approaches that look for products similar to a specification (i.e. requirements or constraints) given by users. McSherry (McSherry 2005) focused on case-based reasoning approaches, in which products are seen as cases from which one should be selected when it is similar to the case (or specification) provided by the user. On each stage of his approach, McSherry asks the user a value for a new product attribute and presents the case that is more similar to the current specification. In addition, users can ask why a certain attribute value is asked, and the approach answers it based on candidate cases, indicating which cases would be selected or rejected based on the user answer. Moreover, in certain cases, the answer might lead to the termination of the dialog.

Another direction is explanation interfaces, which were investigated by Pu and Chen (Pu and Chen 2007). These interfaces organise recommended products in a way that make trade-off situations to be resolved explicit for users, thus facilitating the decision making process. For example, in systems that recommend digital cameras, a group of recommended products can be classified as "cheaper, but with lower resolution."

Recommender systems have been significantly improved by having explanations incorporated into recommendations, but explanation approaches are attached to the approach used to recommend products and do not produce arguments that people adopt to choose or reject options. Stating that “someone like you chose this product” or “you like similar products” is not enough to justify a recommendation. The attachment between the recommendation approaches and explanation types, which typically have an one-to-one mapping, has been criticised by Papadimitriou et al. (Papadimitriou et al. 2011), and they proposed a new taxonomy for explanation types that is based on humans and (product) features. However, the new taxonomy classifies explanations in a different way, but does not solve the problem of producing explanations that only justify the recommendation and not giving the kinds of arguments that users are expecting.

Decoupling the recommendation approach from the explanation was investigated by Zanker and Ninaus (Zanker and Ninaus 2010). They argue that current recommender technology cannot be easily used for explaining why an item is proposed, because the semantics of the model gets lost when factorising and rotating matrices, which are the ideas behind recommender approaches. On the other hand, knowledge-based mechanisms that are associated with a transparent line of reasoning typically do not reach comparable accuracy levels as collaborative mechanisms do. So, the authors propose to decouple reasoning for recommendations from generating explanations. Their approach relies on a layered directed acyclic graph, whose certain paths from a start node to an end node constitute explanations. Even though the idea behind their approach (i.e. decoupling the recommendation from the explanation) is relevant, their briefly described proposal seems to be untractable as the graph is composed of all possible values of the product features (or attributes), and no further discussion is made in this direction.

Finally, content-based and collaborative-based approaches have explored application domains that involve products that people consume in a regular basis, such as movies, books, music, restaurants and news. However, they might be not suitable for the scenario we are exploring, in which people can express preferences for a decision that is not necessarily so frequently performed and therefore there might be not enough data to extract recommendations. Moreover, as Zanker and Ninaus discussed, the models relying recommender system approaches allow only explaining *how* a recommended items is chosen, but not *why*.

7.3 Explanations for Over-constrained Problems

A particular class of decision problems is Constraint Satisfaction Problems (CSPs), which involve imposing a set of hard constraints, i.e. constraints that must be satisfied, and identifying a solution that satisfies all constraints. In situations that this is not possible, that is, the problem is over-constrained, one or more constraints that conflict with each other must be relaxed in order to find a solution for the problem. When a solution is automatically chosen for an over-constrained problem, this solution must be presented for users together with an explanation that justifies why a particular set of constraints was chosen to be relaxed. However, over-constrained problems can have an exponential number of conflicts, which explain the failure, and an exponential number of relaxations, which restore the consistency. So, in this context, there are two main issues: how to choose a relaxation, and how to perform this in an efficient way.

Junker (Junker 2004) has tackled this problem by defining preferred explanations and relaxations based on user preferences between constraints and computing them with a generic method, which not only works for CSPs, but also for any satisfiability problem such as propositional satisfiability or the satisfiability of concepts in description logic. The relaxation problem is defined as $\mathcal{P} := (\mathcal{B}, C)$, where $\mathcal{B}$ is the background containing the constraints that cannot be relaxed, and C is a set of constraints. A subset $\mathcal{R}$ of C is a relaxation of a problem $\mathcal{P}$ if and only if $\mathcal{B} \cup \mathcal{R}$ has a solution. In order to choose a relaxation, Junker assumes the existence of a strict partial order between the constraints of C, denoted by $\succ$, as a user typically prefers to keep the important constraints and to relax less important ones. As $\succ$ is an incomplete specification of ranking among constraints, three extensions are defined in order to produce a total order among constraints, and based on this total order, some conflicts are more relevant for the user than other conflicts. Because of that, Junker is able to define *preferred relaxations* and the analogous definition for *preferred conflicts*. An algorithm based on a divide-and-conquer approach, named *QuickXplain*, is proposed, which has a polynomial response time for polynomial CSPs. Other works have followed the ideas proposed by Junker, and proposed new approaches that aim at optimising the conflict detection in CSPs, such as the *FastXplain* (Schubert et al. 2010).

O'Sullivan et al. (O'Sullivan et al. 2007) argue that many other existing approaches to explanation generation in constraint-based settings are based on the notion of a (setwise) minimal set of unsatisfiable constraints, also known as a minimal conflict set of constraints. However, explaining a relaxation by stating that a conflict is minimal can be not intuitive for users, spurious or misleading (Friedrich 2004), and therefore users need more than one explanation in order to avoid drawing false conclusions. In order to produce better explanations, the notion of *representative set of explanations* is defined, which means that every constraint that can be satisfied is shown in a relaxation and every constraint that must be excluded is shown in an exclusion set. For finding these representative explanations, the algorithm named *RepresentativeXplain* was proposed, which was evaluated in random and real world scenarios.

These works focus on a particular (but relevant) problem, but they are not sufficient for producing explanations for our problem, as it is not a CSP, and involves many other types of preferences besides constraints. In addition, choosing a minimal or less preferred set of constraints to relax can be the wrong choice in many cases, as users may prefer to relax many constraints and do not compromise too much few of them.

7.4 Explanation for Multi-attribute Preference Models

Some approaches have focused on explanations for Multi-Attribute Utility Theory (MAUT), considering utility functions for individual attributes and weights used to sum them, which represent the trade-off among these attributes. Klein and Shortliffe (Klein and Shortliffe 1994) have proposed the *Interpretive Value Analysis (IVA)*, which is a framework for explaining and refining multiattribute value functions automatically. The main idea of the explanation relies on two main concepts: (i) COMPELLINGNESS, which is the weighted difference between the values of a particular attribute (the approach actually considers objectives, which are represented in a tree, in which upper levels are higher-abstraction objectives) of two options and represents how strong an attribute is; and (ii) NOTABLY-COMPELLING, which evaluates if the COMPELLINGNESS is higher than k standard deviations above the average of the values of a particular attribute — "k" determines the degree to which the magnitude of the COMPELLINGNESS of a particular attribute must be an outlier to be considered NOTABLY-COMPELLING with respect to an option, and this parameter might be adjusted for particular users. Based on these two concepts, an explanation is built containing only attributes that are relevant, i.e. when they are "notably compelling." Moreover, Klein and Shortliffe also propose a way of translating all the numerical analysis to make a decision to natural language, so that a user that wants a detailed explanation for the decision can understand the reasoning process. We show below two explanations following this approach, which justify why an option is better than another. The first indicates that only the best of the two options has compelling attributes, while in the second both options have (but in the end SHELL.B is better than SHELL.C). Explanation variables are in *italics*.

– *Price* provides the most compelling reason for the choice.

– *Quality of documentation and quality of front end* are reasons to prefer *SHELL.B* over *SHELL.C*. *Reliability, interactive development facilities, and syntactic familiarity to data processing programmers* are reasons to prefer *SHELL.C* over *SHELL.B*.

A computational model, named Generator of Evaluative Arguments (GEA), was proposed by Carenini and Moore (Carenini and Moore 2006) in order to cover all aspects of generating evaluative arguments in a principled way, by effectively integrating general principles and techniques from argumentation theory and computational linguistics. The approach assumes a previously elicited additive multi-attribute value function (AMVF), which captures user preferences, and based on it, evaluative arguments are generated to support (or not) an option. The argument generation process is divided into two parts — the first consists of selecting arguments to be presented and the second transforms these abstract selected arguments into natural language. As particularities of text generation in natural language is not our concern, we will focus on the first part. Carenini and Moore give seven guidelines to select and produce arguments, such as how to distinguish supporting and opposing arguments based on AMVFs and when an argument is *compelling* (adopted from Klein and Shortliffe's work), and these guidelines are used to create an argumentation strategy, which generates the input of a text generation technique. The approach was evaluated empirically, by testing if tailoring an evaluative argument to the user preferences increases its effectiveness, and if differences in conciseness significantly influence argument effectiveness. While the second hypothesis was confirmed in the experiment, the first one was only marginally confirmed. Note that the most important part to generate explanations rely on Klein and Shortliffe's work.

An approach for selecting and generating arguments for the family of multi-attribute decision models parameterised by weights assigned to the criteria, such as expected utility model and the weighted majority model, was proposed by Labreuche (Labreuche 2011). His approach is based on the analysis of the values of the weights together with the relative scores of the options to be compared. For generating explanations, Labreuche proposes a set of *anchors*, which are a generic way of reasoning in the explanation and look for some changes in the weight vector v that yield an inversion of the prescription made by the decision model. The explanation focuses then on the criteria for which the weight vector has changed. Two strategies for the modification of the weights are considered: the replacement of v by some reference weights $w^{\mathcal{F}}$, and a permutation of the weights v among the criteria (which is associated with a branch-and-bound algorithm). A trivial anchor addresses the case of domination, and another last anchor covers the remaining cases. Examples of each anchor are presented in Table 7.3.

These approaches provide a substantial work on explanation generation for decision making models, with different types of arguments. Nevertheless, they still present limitations on generated explanations, as they do not cover many arguments that users adopt, such as the existence of cut-off values. In addition, there are

Table 7.3: Labreuche's Approach Examples.

Anchor "all"
y is preferred to *x* since *y* is better than *x* on ALL criteria.
Anchor "not on average"
Even though *x* is better than *y* on average, *y* is preferred to *x* since *y* is better than *x* on the criteria *selected pros* that are important whereas *y* is worse than *x* on the criteria *selected cons* that are not important.
Anchor "invert"
y is preferred to *x* since *y* is better than *x* on the criteria *selected pros* that are important and on the criteria *selected pros* that are relatively important, *x* is better than *y* on the criteria *selected cons* that are not important and on the criteria *selected cons* that are not really important, and [criterion *j* for which *y* is better than *x* is more important than criterion *i* for which *y* is worse than *x*] *for all* (i, j) *selected*.
Anchor "remaining case"
y is preferred to *x* since the intensity of preference *y* over *x* on *pros* is significantly larger than the intensity of preference of *x* over *y* on *cons*[, and all the criteria have more or less the same weights].

two main approaches for selecting arguments to be presented in the explanation (decisive criteria) — the use of a threshold (COMPELLINGNESS) and weight analysis — however, there is no consensus which of them (if any) selects the decisive criteria.

7.5 Final Remarks

In this chapter, we presented a review of works that propose approaches for generating explanations that justify decisions made by a software system. The need for explanations emerged from Expert Systems (ESs), which aim at supporting humans to make decisions and need explanations mainly for critical domains, such as medical diagnosis. Even though approaches for ESs advanced this research area, they rely on a huge set of rules that capture domain knowledge, and the bootle neck of this kind of system is still the knowledge engineering. Recommender systems, sometimes seen as the successors of ESs, are now incorporating explanations in their recommendations, but they do not provide explanations for and against recommended products, but only make the recommendation process more transparent for users. This kind of explanation increases the acceptability of recommender systems, but in many cases are not helpful for users to make a decision. In the same way, explanations are produced for justifying relaxations of over-constrained problems; however, this kind of explanation has some of the problems of recommender systems.

Promising, and more general, approaches have been proposed for explaining the result of multi-attribute preference models, using weights to specify trade-off among attributes. These approaches produced substantial results in the explanation generation. However, there are still many challenges to be overcome to produce explanations that support people to make decisions — such challenges include a deep investigation of the kinds of arguments that are helpful, which will be investigated in the next chapter.