Research Issues in Knowledge-Based Configuration

Abstract

Knowledge-based configuration systems have made their way into industrial practice. Nowadays, all major vendors of configuration systems rely on some form of declarative knowledge representation and intelligent search techniques for solving the core configuration problem, due to the inherent advantages of that technology: On the one hand, changes in the business logic (configuration rules) can be accomplished more easily because of the declarative and modular nature of the knowledge bases while on the other hand highly-optimized, domain independent problem solving algorithms are available for the task of constructing valid configurations.

Still, the development has not come to an end as - in a world that becomes increasingly automated and wired together - constantly new challenges for the development of intelligent configuration systems come in: Web-based configurators are being made available for large heterogeneous user groups, the provision of mass-customized products requires the integration of companies along a supply chain, configuration and reconfiguration of services becomes an increasingly important issue, just to name a few.

This chapter gives an overview on these current and future research issues in the domain of knowledge-based configuration technology and thus summarizes the state of the art, recent achievements, novel approaches, and open challenges in the field.

Keywords: Knowledge-based systems, Product configuration systems, Configurator

INTRODUCTION

One of the earliest and most successful expert systems introduced in an industrial environment was a product configurator, when - in the early 1980's - Digital Corp. developed the R1/XCON (McDermott, 1982) system for automating the configuration process for their complex computer systems. Although R1/XCON was one of the first systems of that kind, two typical aspects in the context of configuration systems have not changed since then.

(a) It has been proven that using an intelligent product configurator will lead to significant business benefits: Suitable configurations and accompanying offers can be calculated much faster, the quality of the configurations is comparable or better than the one of manually-engineered solutions, and the process itself is less error-prone which in turn leads to considerable savings for a company (McDermott, 1982; Barker et al., 1989).

(b) There is also another side of the medal, which is for instance documented in (Barker et al., 1989) – also for the R1/XCON system: The configuration task itself can become very complex and the corresponding knowledge bases soon have to contain information on thousands of components and configuration rules, i.e., after ten years of production, the R1/XCON system contained around 10.000 configuration rules. This in turn leads to different problems. So, for instance, maintenance of the knowledge base becomes an issue, in particular in domains where product life cycles are short and changes in the products are frequent. In addition, when the knowledge bases grow, also the running times for checking or constructing a configuration can significantly increase, potentially resulting in performance problems. Finally – as also mentioned already in McDermott (1982) – for all

of the engineering tasks, highly-skilled and trained development staff is needed for maintaining the knowledge bases and/or improving the configurator software.

Due to the inherent complexity of the task, configuration problems have since then always been subjects of interest for researchers in different areas, in particular in the field of Artificial Intelligence (AI). In fact, significant advances have been made since these early, "rule-based" years: Powerful knowledge representation schemes for configuration problems have been developed (Mittal, S. & Frayman, F., 1989; McGuiness & Wright, 1998), a formalization of the problem has been done (Felfernig et al., 2004), the invention of new algorithms was driven by the challenges of the domain (Mittal & Falkenhainer, 1990; Fleischanderl et al., 1988), industrial-strength software libraries are now available (Mailharro, 1998) and knowledge based configurators are nowadays already incorporated in standard business software, see for instance (Haag, 1998).

Nonetheless, the developments of today's networked economy constantly bring in new challenges and requirements for current and future product configuration systems. So for instance the life-cycles of e.g., electronic products still continue to become shorter and shorter while on the other hand the products tend to be more complex which in turn requires even better knowledge-representation and modelling schemes for alleviating the knowledge-engineering and maintenance tasks. In addition, with the growing complexity of the knowledge-bases, we also need adequate means and tools to support the error detection and debugging phases, since standard debugging techniques are insufficient for such knowledge-based systems.

When looking at current developments from the business perspective, we can observe a trend that companies today aim at co-operating in supply-chain networks in the process of manufacturing and provisioning their configurable goods and services. So in many cases, the customer requirements that determine the design of the final product have to be forwarded to the partners in the supply-chain while, at on the other hand, overall consistency of the final configuration has to be ensured.

Another business-related aspect in some domains can be seen in the fact that sometimes a fully configured product which has been already deployed at the customer's site for years, e.g., a large telecommunication switch (Fleischanderl et al., 1988), has to be extended or adapted to reflect new customer requirements. In these cases, the existing configuration has to be reconfigured and we have to consider different optimization goals, like for instance preserving most of the existing configuration.

An additional example where new requirements for product configuration systems can be seen is the fact that nowadays not only technical engineers or sales engineers are the users of configuration systems as it was in the early years. The Mass Customization paradigm is applied to various business branches, and in many domains the customers themselves use the configurator (over the Web) to tailor a product to their specific needs and demands. This in turn creates new challenging requirements on interactivity for configurators: End-users can be quite heterogeneous with respect to their background and capabilities, so things have to be simple for them. On the other hand, users may have different preferences and capabilities how they want to or are able to express their requirements. As such, a personalized and more intelligent user interface may be required to guarantee the success of an online configuration system.

This chapter gives on overview on such current and future research issues for intelligent product configuration systems and thus summarizes the state of the art, recent achievements, novel approaches, and open challenges in the field. Quite naturally of course, the selection of issues is determined by a somehow subjective viewpoint, but it is

based both on long years of active research in the field as well as on the experiences and lessons learned from international workshops that have been held in the last decade on major Artificial Intelligence conferences.

KNOWLEDGE REPRESENTATION AND REASONING

Successful applications of configuration technologies can be found in various domains such as the automotive industry (Freuder et al., 2001), the telecommunication industry (Juengst & Heinrich, 1998), the computer industry (Barker et al., 1982) or power electric transformers (McGuiness & Wright, 1998a). *Rule-based* knowledge representations, such as used in R1/XCON (Barker et al., 1982), were the starting point for configuration knowledge representation. In later years, *model-based* knowledge representations have been developed which strictly separate domain knowledge from problem solving knowledge: Such a separation increases the effectiveness of configuration application development and maintenance (Juengst & Heinrich, 1998; Freuder et al, 2001; Forza & Salvador, 2002; Mailharro, 1998) since changes in the product knowledge do not effect the definition of the search process and vice versa. Overall, a comprehensive configuration environment (Mittal & Falkenhainer, 1990) has to support all major tasks of *core configuration,* i.e., guiding the user and checking the consistency of user requirements with the knowledge-base, solution presentation, and translation of configuration results into detailed bill-of-materials.

Typically, knowledge-bases are built using proprietary languages, see, e.g., (Franke & Piller, 2002; Haag, 1998; Forza & Salvador, 2002), where technical experts and knowledge engineers elicit product, marketing and sales knowledge from domain experts. Knowledge bases thus consist of a description of the product structure and a set of constraints that restrict the combinations of components in a configuration result. Configuration problem solving is in many cases based on a *Constraint Satisfaction Problem* representation of a configuration task (Tsang, 1993). Depending on the size and complexity of the problem, different facets of constraint representations can be applied: In *Generative Constraint Satisfaction* (Fleichanderl et al., 1998) components are dynamically generated on demand during the search process; when using a *Dynamic Constraint Satisfaction* (Mittal & Falkenhainer, 1990) approach, depending on a specific state in the search process only a relevant subset of the defined constraints and variables are active, i.e., are taken into account for calculating a solution; in *Distributed Constraint Satisfaction* (Yokoo et al., 1998), messages about changes in the problem space are exchanged between different local entities of constraint satisfaction problems. Such a distributed representation can primarily be applied in different variants of supply chain settings.

Although configuration systems have been successfully applied in various real-world applications, a number of challenges has to be tackled with respect to configuration knowledge representation: Knowledge base development is a cooperative process between technical experts and domain experts and the development of knowledge-bases can be very expensive (Mittal & Frayman, 1989). In this context, the application of *standard representations* can help to reduce development and maintenance costs because standards are known by technical experts and in many cases are also known by domain experts (generally non-programmers). Furthermore, information systems departments always aim at standardization and *interoperability* between various system components. Therefore, configuration systems are required to be equipped with standard representations which contribute to an improved flexibility of a company’s software infrastructure: In the financial services domain, for instance, standardized interfaces are a major decision

criterion for incorporating a configurator into their software environment. The amount of resources required to develop and maintain configuration knowledge-bases can be substantial (see, e.g. Mittal & Frayman, 1998). In many cases, however, a configuration knowledge base is encoded in the proprietary language of the underlying configuration environment. This makes related investments particularly unsafe due to the fact that, e.g., changing requirements on the configurator application could lead to a need of exchanging the whole configuration environment (Mittal & Frayman, 1998). In such cases, no support will be available for easily transforming an existing knowledge-base into the representation of the new environment. Therefore, the following languages can play an important role in the context of standardized configuration knowledge and product data representation.

OIL and *DAML+OIL* (Fensel et al., 2001) are ontology representation languages developed within the context of the *Semantic Web* (Berners-Lee, 2001). These languages support the design of ontologies on the formal basis of description logics. Felfernig et al. (2003) point out that Semantic Web representation languages are suitable for configuration knowledge representation. However an additional language is needed supporting an intuitive formulation of constraints on product structures, especially the definition of aggregation functions and complex structural properties is not supported by state-of-the-art Semantic Web knowledge representation languages. With respect to ongoing efforts to extend DAML+OIL or its successor *OWL* (van Harmelen et al., 2001), the work of Felfernig et al. (2003) contributes a set of criteria which must be fulfilled in order to apply those languages for full-fledged configuration knowledge representation.

Universal Standard Products and Services Classification Code (*UNSPSC*) is a coding system organized as product taxonomy. Levels of the taxonomy are *segments* denoting aggregations of families (e.g., computer equipment), *families* as groups of interrelated categories (e.g., software), *classes* as a group of elements sharing a common usage (e.g., text-editing), and *commodity* as a group of substitutable products (e.g., Linux text editors). *RosettaNet* classification schemes are restricted to the categorization of electronic equipment. RosettaNet has two taxonomy levels (product groups and products). Both standards focus on the categorization of products but do not provide mechanisms for building models of generic product structures. Another standard related to product data representation is cXML (*commerce XML* - www.cxml.org) which as well does not provide any mechanisms for configuration knowledge representation (Schmitz et al., 2004).

The *standard for product model interchange* (*STEP* – ISO, 1994) takes into account all aspects of a product including geometry and even organizational data. The goal is to provide means for defining application specific concepts for modeling products in a particular product domain. Such application specific concepts are denoted as *application protocols*, which are defined using the *EXPRESS* DDL. EXPRESS includes a set of modeling concepts useful for representing configurable products it can, however, not be used to define enterprise-specific configuration models without leaving the STEP standard: The reason is that STEP standards define a (although generic) fixed product structure, i.e., they do not provide the freedom to design any type of configuration model. If a company models its products according to STEP, it should use an application protocol in order to conform to the STEP standard.

The *Unified Modeling Language* (*UML*) and the *Object Constraint Language* (*OCL*) (Warmer & Kleppe, 2003) include major language elements needed for the intuitive representation of configuration knowledge (Felfernig et al., 2002). Such a standardized language is a crucial success factor for integrating configuration technologies into industrial software development processes. Object-oriented structure representation concepts of UML (Dennis et al., 2004; Rumbaugh et al., 1989) and OCL Constraint

Definitions allow the representation of configuration knowledge in a quite natural way: Product components are represented as classes and constraints between different components are represented by a set of corresponding OCL navigation expressions, i.e. the basic concepts provided by UML/OCL should be integrated into existing configuration environments. From the viewpoint of knowledge acquisition support for configuration knowledge bases, the integration of industrial standard representations such as UML/OCL into configurator development environments is one of the major challenges for allowing effective knowledge acquisition processes, exchangeability of knowledge bases, and standardized interfaces to existing software components.

DEVELOPMENT AND DEBUGGING SUPPORT

Effective knowledge acquisition and maintenance support is one of the key issues in building configuration knowledge bases. The application of, e.g., UML/OCL for configuration knowledge representation (Felfernig et al., 2002) can be seen as a quite intuitive and understandable representation of configuration knowledge. However, the application of such modeling languages for knowledge base construction does not automatically guarantee the *validity* of the generated knowledge base: Validation is typically performed by testing the knowledge base using a set of *test cases* (*examples*) which are provided by domain experts or are taken from previously calculated configurations. If a knowledge base is invalid, i.e., some of the provided test cases are inconsistent with the actual version of the knowledge base, domain experts and knowledge engineers are faced with the challenge to identify a set of constraints in the knowledge base which are responsible for the faulty behavior of the knowledge base. Such a task becomes even harder when applying *rule-based* knowledge representations: In such a situation, domain experts and knowledge engineers need adequate tools that support automated identification of the faulty parts in a configuration knowledge base. *Automated debugging* concepts can improve the effectiveness of configuration knowledge base development processes by significantly reducing development efforts: If a new version of a configuration knowledge base is created, regression tests can be automatically triggered in order to assure that the new version is still consistent with the defined test cases. Automated regression tests reduce development and maintenance costs since faults in the knowledge base can be detected early, i.e. they are not propagated to the production environment.

In order to support an automated detection of faulty constraints in configuration knowledge bases we can apply concepts from *model-based diagnosis* (Reiter, 1987): Model-based diagnosis is characterized as explanation of faulty behavior based on observations of the behavior of the concrete system (e.g. the behavior of a mal-functioning car engine) and the comparison with a corresponding system model representing the correct behavior of the system. The theory of Reiter (1987) includes the basic representational and computational assumptions which can be applied to a number of application domains. In Reiter (1987), a *system* is interpreted as a pair (SD, COMPS) where SD, the system description, is a set of first-order sentences and COMPS, the system components, is a finite set of constants. An observation, OBS, of a system is a finite set of first-order sentences. Thus (SD, COMPS, OBS) denotes a system (SD, COMPS) with observations OBS. A corresponding diagnosis for (SD, COMPS, OBS) is a minimal set $\Delta \subseteq$ COMPS such that SD $\cup$ OBS $\cup$ {ab(c)| c $\in$ Δ} $\cup$ {$\neg$ab(c)| c $\in$ COMPS - Δ} is consistent. In other words, the assumption that the components of {ab(c)| c $\in$ Δ} behave abnormal together with the assumption that the other components (i.e., {$\neg$ab(c)| c $\in$ COMPS - Δ}) behave normal, is consistent with the given system description SD and the given observations OBS.

If we now interpret the logical sentences (constraints) of a configuration knowledge base as system components, we can introduce a *configuration knowledge base* (*CKB*) *diagnosis* problem and a corresponding *configuration knowledge base diagnosis*. A detailed discussion on the application of model-based diagnosis concepts to the automated debugging of configuration knowledge bases can be found in (Felfernig et al., 2004): A CKB Diagnosis Problem is a triple (DD, E^+, E^-), where DD is a configuration knowledge base, E^+ is a set of *positive examples* (test cases), and E^- is a set of *negative examples* (test cases). The examples are given as sets of logical sentences. Each example on its own is assumed to be consistent. Note that a positive test case is an example for the intended behavior of a knowledge base; a negative test case is an example for an unintended behavior of a knowledge base (i.e. an example for a result which should not be calculated by the knowledge base).

A CKB Diagnosis for a CKB Diagnosis Problem is a set $\Delta \subseteq$ DD of sentences such that there exists an extension EX (EX is a set of logical sentences) such that DD-$\Delta \cup$ EX $\cup$ e^+ is consistent $\forall$ $e^+ \in E^+$, and DD-$\Delta \cup$ EX $\cup$ e^- is inconsistent $\forall$ $e^- \in E^-$. The result of a diagnosis task is a (minimal) set of constraints of the configuration knowledge base which should be taken into account in order to make all positive examples consistent with the knowledge base.

This process of a consistency-based diagnosis of a configuration knowledge base can be fully automated given a number of pre-defined test cases. The definition of test cases is an important precondition for the application of model-based diagnosis concepts: On the one hand, test cases can be derived from former valid configuration results. If such results are not available, test cases must be either manually specified by domain experts or can be (semi-)automatically generated from the specification of a configuration knowledge base. Although testing is considered the most pragmatic and successful technique in quality assurance, the research field is still insufficiently explored in the context of developing and maintaining knowledge-based systems (Preece et al., 1997; Pretschner, 2001). The development of intelligent concepts supporting the automated generation of test suites should therefore be one of the major focuses of current research. In that context, the major challenge with respect to validation of configuration knowledge bases is the development of intelligent concepts that support the *generation and administration of test cases*. Test cases can be automatically generated from a given definition of a configuration knowledge base by calculating possible user interactions (requirements specifications) for predefined interaction paths. The number of generated test cases has to be further reduced in order to make their validation feasible for domain experts within a reasonable time span Felfernig et al., 2005).

DISTRIBUTED CONFIGURATION

Most of the existing approaches in the area of knowledge-based product configuration rely on the assumption that the configurable product is provided by *one single supplier* who assembles the product according to the customers' needs. However, in today's networked economy, the provision of products and services in many domains requires the integration of different companies in a supply-chain. Furthermore, in many cases the sub-assemblies are again designed to be configurable and the detailed configuration of the sub-assembly partially depends on the configuration parameters of the product as a whole. The standard approach to solve such problems is to integrate the required configuration rules into a central knowledge base. While this seems intuitive and practicable at a first glance, such an approach may have the drawback that the resulting knowledge base will soon become

complex, thus increasing the probability of errors and consequently maintenance costs. In addition, it may be undesirable for a supplier to expose all its configuration logic - which is in many cases connected to a company's pricing rules - for confidentiality or organizational reasons.

In the CAWICOMS project (Ardissono et al., 2000, Ardissono et al., 2003), a consortium strongly rooted in telecommunications industry was piloting a scenario of cooperating configuration systems in the domain of IP-based Virtual Private Networks (IP-VPN). In that business case, a reseller is contracting multi-national IP-VPN solutions to its clients: The reseller subcontracts parts of the network to different telecommunication service providers that might themselves introduce sub-suppliers. In order to generate a full configuration of the overall solution including e.g. IP-settings, internal routings and additional hardware requirements, the cooperation of the local problem solvers of the involved business entities is necessary. Thus, the value chain has a networked structure, where each node can be represented by a configuration agent; in addition, *mediating components* were introduced for the co-ordination of the different configuration systems.

The main findings of the project with regard to distributed configuration and issues for future work in the area can be summarized as follows.

(a) The interoperation of different configuration systems requires the establishment of a shared view on the interdependent fractions of the product model. Therefore, a common language and ontology has to be employed for describing the configurable product configuration problem in a tool-independent manner. A proposal for such a common knowledge representation mechanism and exchange format based on a general configuration ontology in the sense of (Felfernig et al., 2002) has been developed within the project. However – as already sketched in the section on knowledge representation – up to now there exists not yet a *lingua franca* for exchanging configuration knowledge, which is one of the major prerequisites needed in such distributed configuration scenarios.

(b) Standard algorithms for distributed solution search like Distributed Constraint Satisfaction (Yokoo, 2001) are not directly applicable for distributed configuration problems. Thus, domain-specific distributed algorithms, see e.g. (Ardissono et al., 2003; Zanker, 2002) which address the particularities of the problem domain have been designed and prototypically implemented in the CAWICOMS project. However, there is still room for improvement to ease the integration of distributed solving protocols with the different heterogeneous configuration systems on the market.

RECONFIGURATION

Nearly all of the existing product configuration frameworks, problem solving algorithms, and configurator applications are designed for the use in business scenarios in which a customer-specific product variant is constructed "from scratch" (Männistö et al., 1999).

Still, there are many domains in which the products or services have a longer life time and for which it is common that the original configuration has to be adapted or extended over time. A typical example can for instance be found in the domain of the configuration of large telecommunication switches, see, e.g. (Fleischanderl et al., 1998): These large-scale complex electronic devices are configured once and set up at a location and are then in productive years for several years or even decades. During this period of productive use, there might be several reasons, why the original configuration has to be changed or extended: For instance, the requirements may change over time when new technologies

become available or more switching-capacity has to be provided. On the other hand, it is also possible that individual broken components have to be replaced and only newer versions of these components are available which have to be plugged into an existing configuration.

Quite obviously, the optimization goals for configuration from scratch and reconfiguration are not the same: While in an initial configuration the main focus will be in general on minimizing the number of components needed in the system, the reconfiguration goal in most cases is to preserve as many parts of the existing configuration as possible. Alone when considering these two different goals of optimization, we see that a system which is designed to support *reconfiguration* has to be augmented with additional knowledge, configuration logic, or adequate heuristics beside the core business and configuration rules.

In principle, two different approaches for dealing with that challenge can be identified: First, the required reconfiguration knowledge can be modelled externally as an add-on to the existing knowledge base. Such an approach is for instance proposed in (Männistö et al., 1999), where reconfiguration *operations* consisting of pre-conditions and actions are made explicit in a separate “reconfiguration model”. Given an existing configuration and a set of new requirements, the reconfiguration problem then basically consists of finding a subset of these reconfiguration operations that change the system in a way that the new requirements are fulfilled. The main advantage of such an approach lies in the fact that the search space for possible modifications is limited by the number of existing reconfiguration operations: If we assume that none of the reconfiguration operations will lead to a violation of the original configuration constraints, the problem variables of the typically larger core configuration problem do not have to be taken into account during search. The main drawback of such an approach however, can be seen in a) the problem of ensuring that particular consistency property for complex knowledge bases and b) that the reconfiguration rules have to be maintained and updated every time the core configuration knowledge base is changed.

The other principle approach is to include all reconfiguration alternatives and –knowledge in the knowledge base from the beginning and only change the optimization goal when it comes to reconfiguration such that configurations which re-use more of the existing components are preferred over others. While such an approach in theory will lead to "optimal" reconfigurations and no additional modelling is required, the search problem is in practice intractable with today's search technology for realistic scenarios. In fact, in many cases even the original configuration may already be suboptimal due to the complexity of the search space and the use of domain-specific heuristics (Fleischanderl et al., 1998) which are needed to find a *good* solution in an appropriate time frame. In addition, many problem solvers (e.g., based on Constraint Satisfaction) are based on *Constructive Search and Backtracking*, i.e., the configurations are incrementally constructed from scratch during the search process. Therefore, in our opinion, *local search techniques* or *evolutionary* algorithms are more promising for reconfiguration problems and should be further investigated in future research: As a core characteristic, these algorithms start from an existing configuration (or variable assignment) and incrementally try to improve the current solution by exploring *neighbouring* solutions which has a strong correlation to the reconfiguration problem.

A special form of reconfiguration support can be seen in what is called *Parametric Re-design*: Such approaches are basically suitable for configurable systems that are already "designed for reconfiguration". In such systems, the basic structure of the product remains static (e.g., there exists a given number of connected components), but the individual components can be parameterized such that they fulfil specific requirements. In (Stumptner

& Wotawa, 1999), for instance, such an approach is described for the domain of telephone networks: Given an existing network configuration and a new functional requirement (in that case a certain call type) the problem is to reconfigure the nodes in the telephone network such that this functionality becomes available. In their approach, Stumptner and Wotawa propose to employ model-based diagnosis techniques for determining these parameter sets and thus develop a general framework for parametric reconfiguration. The search space in their approach remains manageable due to the fact that they limit the reconfiguration options to alternative parameter settings. Model-based diagnosis techniques are also used for reconfiguration in the approach proposed by Crow & Rushby (1991): In contrast to the work of Stumptner and Wotawa, however, they base their approach on explicit reconfiguration knowledge. Their main goal was to extend existing diagnosis techniques toward automatic *repair*, where the goal not only is to identify faulty components of a system but also compute a set of actions to be taken in order to re-establish a functioning system.

From a business perspective, re-configuration (adaptation/extension) of already installed systems falls into the category of *after-sales* and *maintenance* activities, which we see as an increasingly important business area for many of today's companies (Männistö et al., 1999). As such, we argue that adequate tool support for such high-quality customer services can be extremely valuable for companies, in particular as a competition factor in markets where the products of different manufacturers are comparable and differentiation from the market competitors has to be achieved by the provision of such *add-on* services.

In the context of reconfiguration and after sales services, we also see *versioning* and *evolution* of configuration knowledge bases as additional, future challenges in the area of knowledge-based configuration, in particular, as there exists nearly no support in current configurator systems.

PERSONALIZED USER INTERACTION

In Business-to-Consumer environments, product configuration systems are one of the information systems that are directly located at the interface between customers and producers. They allow the automation of the order taking process by capturing customer requirements without involving human intermediaries in an interactive process. This means that customers are enabled to self-configure their products to their individual requirements in web-based applications, which has been proven to lead to a significant reduction of costs and errors. Thus, product configuration systems are one of the main enablers of the Mass Customization paradigm which aims at providing highly tailored products and services to end users.

While technical experts have been the dominant user group of configuration systems in the early years, nowadays they are used by quite heterogeneous groups of online end users. Thus, these users have typically different backgrounds in terms of experience or skills ore are simply different in the way they prefer to or are able to express their needs and requirements.

Consequently, it is obvious that static “one-style-fits-all” approaches are not adequate for user interfaces of configuration systems in many of current application environments. In web-based applications, for instance, HTML forms, with which the user can specify technical product details in order to find a valid product configuration, are common. However, such static state-of-the-art technologies can not ensure that the final configured product is tailored to the *real* customer preferences and expectations. Thus, the quality of

the achieved results can be significantly improved when the system interacts with the user in a personalized way.

Up to now, there have been significant efforts for personalizing web-based user interfaces of configuration systems. Besides “standard” personalization techniques for hypermedia applications (Kobsa et al., 2001), there are several approaches that are specific for personalization in the domain of configuration systems. In general, we can distinguish two basic strategies how user requirements can be elicited: (a) The system poses explicit questions to the user; (b) the system applies indirect reasoning techniques about, for instance, already gathered information about the current user or some stereotype classification mechanisms.

In the CAWICOMS project (Ardissono et al., 2000, Ardissono et al., 2003), such a hybrid approach was chosen for the domain of configuration of complex telecommunication switches. The CAWICOMS system automatically adapts its user interaction to the user’s skills by varying the complexity level of the configuration process based on information about the current user stored in a long-term user model (Thompson et al., 2004); on the other hand, a rule-based system is applied to determine a more “overall” personalization strategy. For instance, in the CAWICOMS system the user can delegate decisions about feature values to the system which infers the most suitable setting. Furthermore, the system hides or presents some details in the configuration process in order to focus the presentation on the most relevant information according to the current user’s skills.

One of the major challenges of a hybrid approach, however, lies in the fine-tuning of the interoperation between the two techniques. The explicit personalization rules in such a system are in many cases driven by the estimates contained in user models based on Bayesian Networks, cf. for instance the POET tool (Royalty et al., 2002). As a side effect, small changes in such a user model can cause a threshold to be exceeded such that an expert rule fires causing unexpected major effects.

Today, systems are wide-spread that base the personalization solely on long-term user models, reasoning on past behaviour, or stereotypes. These systems are producing promising results in particular in domains with re-visiting users, see for instance (Thompson et al., 2004; Sung, 2002). For first-time visitors, however, they typically face the *new user problem*, i.e. in absence of suitable data for a new user they can base their personalization simply on a poor user model. In addition, user models that are maintained by such systems can often only capture high-level characteristics of the customer to be theoretically reusable across different applications.

Therefore, we claim that it is particularly important to consider extensive personalization of the interaction between the user and the configuration system to reach better results and higher customer satisfaction. Thus, it is not sufficient to adapt the content and the presentation of configuration application to the current user’s skills, but also to tailor the interaction level (Jannach & Kreutler, 2005). For example, if we think of a system for configuring personal computers, there will be users who want to specify technical details of the desired model, whereas others will only be able to express for what purposes they intend use the computer; others again only want to compare preconfigured models and decide by themselves. Up to now, such a personalization of the interaction with the user was primarily addressed in the context of recommender systems.

McGinty and Smyth (2002) propose an approach for recommender systems that is based on a more casual conversation. This means that there should be several degrees of feedback that an online user can provide during the dialog. For instance, leading users through deep dialogs that replicate customer buying models from real world is not always

appropriate in online settings. The authors argue that there should also be a low-cost form of feedback for users instead of complex dialogs. They propose a comparison-based approach for product recommendation, in which the user is asked to choose a recommended item as a (positive or negative) preference. The further recommendation is based on the difference of the preferred products and the remaining alternatives.

In a further investigation, McGinty and Smith (2002a) give an overview on different techniques for user feedback which can be based, e.g., on value elicitation, tweaking, ratings, and preferences. In their work, they focus on a low-cost preference-based feedback model which is evaluated in a recommendation framework. The proposed feedback techniques for recommender systems can also be applied in the context of configuration systems. For instance, comparison-based approaches could be applied when the final configurations are presented to the user.

In the domain of configuration systems, Pu et al. (2003) consider preference elicitation as a fundamental problem: Building on experiences in building decision support systems in various domains, they identify some principles for designing of the interactive procedure of finding a suitable configuration in the solution space. In a survey of ten commercial online flight reservation systems, they find out that a personalized user interaction significantly improves the preference elicitation process for the end-user. They allow users to state values for those options that correspond to their main objectives, which leads more quickly to a more accurate preference model. Furthermore, example critiquing in a minimal context, i.e. making critiques on a personalized (minimized) set of attributes, is also identified as adequate means. Finally, the authors also consider the visualization of the result set with the possibility of revising previously stated preferences during the elicitation process as crucial because users can immediately see the consequences of their stated preferences and possible changes.

EXPLANATIONS & PROBLEM RESOLUTION

Most of today's product configuration systems are highly interactive software applications. Typically, users incrementally enter their requirements while the system continuously checks the constraints, eventually reports conflicts or removes inconsistent options. Then, the user may be given the possibility to revise his/her requirements and finally, the system in many cases automatically completes the configuration by, e.g., adding some mandatory components or by setting some variables with default values.

In particular for configuration applications where the end user is not a technical expert, e.g. in a Web-based environment, it is important that the end user develops a good understanding of the behaviour and logic of the configurator. Typical questions that arise in such interactive sessions are, for instance, "*why I am not allowed to select this option anymore?*", "*what decision do I have to retract, if I want to have a particular functionality?*", or "*why was an option automatically set by the configurator with a certain value?*". If the user is not provided with answers to these questions, there is a high chance that the user's confidence in the final configuration is low or he/she is frustrated when using the system because no adequate help was available during the interactive configuration session. Overall, we therefore claim that the provision of explanations can significantly help to increase the acceptance and value of a configuration system.

In fact, the ability to provide explanations can be seen as one of the key features of knowledge-based systems in general and there also exists long research history on how problem solvers can be built that are capable of e.g., justifying conclusions, detecting

inconsistencies, or keeping track of dependencies, see (Forbus & deKleer, 1993) for an overview. Still, many of the proposed approaches like so-called Truth Maintenance Systems are limited in their applicability, since the size of the dependency network which has to be maintained soon gets too large to manage problems of realistic size.

Today, Constraint Satisfaction (Tsang, 1993) is the most popular technique for representing configuration problems. Therefore, most of current research efforts in the area of advanced user interaction and explanations are based on this technology. On the other hand, many researchers also use configuration problems as a test-bed for new algorithms because of the typically high complexity that is involved in solving such problems. A recent contribution in that area was introduced by Junker (2004), who developed QuickXPlain, a general algorithm for fast extraction of dependencies and detection of *minimal* conflicts for arbitrary constraint propagation and inference algorithms, which is in particular of importance because high-performance problem solvers in many cases lack the ability to provide explanations. With the help of QuickXPlain, some of the answers mentioned above (e.g., why a certain value was chosen) can be answered without the need of costly book-keeping of justifications and in cases, where a black-box propagation engine is used that does not record the explanations.

Beside Junker's algorithm also other proposals for computing explanations for Constraint problems have been developed in recent years, see e.g., (Rochart et al., 2003; Freuder et al, 2001; Jussien, 2001). Still, many of them rely on the abduction principle which can lead to the problem of "spurious explanations" which were dealt with by Friedrich (2004). In this paper, the notion of a "well-founded explanation" was introduced which helps us to eliminate such inconsistent explanations.

Conflict detection and explanation of the situation is, however, only one part of the problem. In fact, for the end user, it may be desirable to get some advise how to deal with the problematic situation, i.e., how conflicts can be resolved and consistency can be restored. The work of Amilhastre et al. (2002) is one example of research in that direction. The main problem of consistency restoration in constraint-based configuration problems lies in the fact, that most current approaches for performing the required tasks are computationally expensive while at the same time the response times in an interactive configuration session are very restricted. The authors therefore propose a technique that relies on pre-compilation of the problem, which is in our opinion in general a technique which is also promising for other problem domains that require fast response times. In their work, the original constraint problem is pre-processed offline and an automaton is compiled that represents to set of possible solutions. At run-time, the generated data structures can then be exploited for consistency maintenance and restoration in an efficient way.

Despite these recent advantages, we still can identify additional challenges that have to be addressed in the context of explanations and consistency restoration in interactive configurator applications: Although there are already first commercial solutions that incorporate explanation facilities (like Configurator software), support for *repair* in case of inconsistent situations cannot be found in today's systems. We also think that still better algorithms are required in that context that for instance take the particularities of configuration problems better into account – in contrast to only viewing the configuration problem as a (binary) Constraint Satisfaction Problem.

From our subjective perspective, another challenge in that context lies in the automated construction of *understandable* explanations: If we look at what we can get from current configurator software solutions is in many cases not more than a *trace* of the inferences

(e.g., propagations) that were made by the problem solver. In the best case, such traces are understandable for the knowledge engineer who designed the knowledge base, but not for the end user. Furthermore, we think that there is also an unexploited potential for personalization in explanations: Depending on the goal one wants to achieve with the explanations, for instance, increasing the user's confidence or helping the user to understand the logic behind, the style or technical depth of the explanations could be varied.

CONCLUSIONS

The technological and economical environment in which product configuration systems are embedded is rapidly evolving, which – as a consequence – constantly creates new challenges for today's and future software systems that shall efficiently support the process of configuring products and services according to the customers' needs. Although there exists a long history of applying knowledge-based technology for solving configuration problems, further research in various directions – from knowledge representation, over problem solving, or personalization – will be required to cope with these new requirements. Within this chapter, an overview on these new challenges was given and recent developments and novel approaches in knowledge-based configuration technology were summarized.

REFERENCES

Amilhastre, J., Fargier, & H., Marquis, P. (2002) *Consistency Restoration and explanation in dynamic CSPs - Application to Configuration*. Artificial Intelligence, 135, 199-234.

Ardissono, L., Felfernig, A., Friedrich, G., Goy, A., Jannach, D., Petrone, G., Schäfer, R., & Zanker, M. (2000) *Personalizing on-line configuration of products and services*, In Proceedings of the 15th European Conference on Artificial Intelligence, Lyon, France, IOS Press, 225-229.

Ardissono, L., Felfernig, A., Friedrich, G., Goy, A., Jannach, D., Petrone, G., Schäfer, R., & Zanker, M. (2003) *A Framework for the Development of Personalized, Distributed Web-Based Configuration Systems*, AI Magazine, 24(3), Fall 2003, p. 93-110.

Barker V.E., O'Connor D.E., Bachant J.D., & Soloway E. (1989) *Expert systems for configuration at Digital: XCON and beyond*. Communications of the ACM, 32(3), 298-318.

Berners-Lee T. (2001) *Weaving the Web*. Harper Business.

Crow, J. & Rushby, J. M. (1991) *Model-Based Reconfiguration: Toward an Integration with Diagnosis*, National Conference on Artificial Intelligence – AAAI'91, AAAI Press, 836-841.

Dennis A., Wixom B., &Tegarden D. (2004) *System Analysis and Design with UML Version 2.0: An Object Oriented Approach*. John Wiley & Sons, 2nd edition.

Edwards K. & Pedersen J. (2004) *Product Configuration Systems – Implications for Product Innovation and Development*, In: Proceedings International Conference on Economic, Technical and Organisational aspects of Product Configuration Systems, Kopenhagen, Denmark, 231-239.

Felfernig A., Friedrich G., Jannach D., & Zanker M. (2002) *Configuration Knowledge Representation Using UML/OCL*. In: Proceedings of 5th International Conference on the Unified Modeling Language (UML 2002), Dresden, Germany, 49-62.

Felfernig A., Friedrich G., Jannach D., Stumptner M., & Zanker M. (2003) *Configuration knowledge representations for Semantic Web applications*, Artificial Intelligence in Engineering, Design and Manufacturing, 17(3), 31-49.

Felfernig, A., Friedrich, G., Jannach, D. & Stumptner, M. (2004) *Consistency-based diagnosis of configuration knowledge bases*. Artificial Intelligence, 152(2). 213-234.

Felfernig A., Isak K., & Kruggel T. (2005) *Testing Knowledge-based Recommender Applications*. OEGAI Journal, Special Issue on Recommender Systems, 24(4), 12-18.

Fensel D., van Harmelen F., Horrocks I., McGuinness D., & Patel-Schneider P. (2001) *OIL: An Ontology Infrastructure for the Semantic Web*. IEEE Intelligent Systems, 16(2), 38-45.

Fleischanderl, G., Friedrich, G., Haselböck, A., Schreiner, H., & Stumptner, M. (1998) *Configuring large systems using generative constraint satisfaction*. IEEE Intelligent Systems, July/August 1998, 59-68.

Forza C. & Salvador F. (2002) *Managing for variety in the order acquisition and fulfillment process: The contribution of product configuration systems*. International Journal of Production Economics, Elsevier, Vol. 76, 87-98.

Franke N. & Piller F.T. (2002) *Configuration Toolkits for Mass Customization: Setting a Research Agenda*, Working Paper No. 33 of the Department of General and Industrial Management, Technische Universität München, ISSN 0942-5098.

Friedrich, G. (2004) *Elimination of Spurious Explanations*. In: Proceedings of the 16th European Conference on Artificial Intelligence (ECAI 2004), Valencia, Spain. IOS Press, 813–817.

Forbus, K., & deKleer, J. (1993) *Building Problem Solvers*. MIT Press.

Freuder, E., Likitvivatanavong, C., & Wallace, R. J. (2001) *Deriving explanations and implications for constraint satisfaction problems*, In: Principles and Practice of Constraint Programming - CP 2001, Springer LNCS 2239, Paphos, Cyprus, 585-589.

Haag, A. (1998) *Sales Configuration in Business Processes*. IEEE Intelligent Systems, 13(4), 78-85.

ISO (1994) *Standard 10303-1: Industrial automation systems and integration – Product data representation and exchange – Part 1: Overview and fundamental principles.*

Jannach, D. & Kreutler, G. (2005) *Personalized User Preference Elicitation for e-Services*, In: Proceedings of IEEE International conference on e-Technology, e-Commerce and e-Service, Hong Kong, p. 604-611.

Juengst E.W. & Heinrich M. (1998) *Using Resource Balancing to Configure Modular Systems*. IEEE Intelligent Systems, Special Issue on Configuration, 13(4), 50-58.

Junker U. (2004) *QuickXplain: Preferred Explanations and Relaxations for Over-Constrained Problems.* Proceedings AAAI'2004, San Jose, 2004, pp. 167-172.

Junker U. (2001a) *Preference programming for configuration*. In: Proceedings of Workshop on Configuration (IJCAI'01), Seattle, WA, 50-56.

Jussien, N. (2001) *e-constraints: explanation-based Constraint Programming*, In: CP'01 Workshop on User-Interaction in Constraint Satisfaction.

Kobsa, A, Koenemann, J., & Pohl, W. (2001) *Personalized Hypermedia Presentation Techniques for Improving Online Customer Relationships*, The Knowledge Engineering Review, 16(2), 11-155.

Mailharro, D. (1998) *A classification and constraint-based framework for configuration*. Artificial Intelligence in Engineering, Design and Manufacturing, Vol. 12(4). 383–397.

Männistö, T., Soininen, T. Tiihonen, J., & Sulonen, R. (1999) *Framework and conceptual model for reconfiguration.* In: Papers from the AAAI Configuration Workshop, AAAI Technical Report WS-99/05. AAAI Press.

McDermott, J. (1982) *A Rule-Based Configurer of Computer Systems*. Artificial Intelligence 19 (1), 39-88.

McGuiness, D.L. & Wright, J.R. (1998) *Conceptual modelling for configuration: A description logic-based approach*. Artificial Intelligence in Engineering, Design and Manufacturing (AI EDAM), 12(98), 33–344.

McGuiness D. & Wright J. (1998) *An Industrial Strength Description Logics-Based Configurator Platform*, IEEE Intelligent Systems, Special Issue on Configuration, 13(4), 69-77.

McGinty, L. & Smyth, B. (2002) *Deep Dialogue vs. Casual Conversation in Recommender Systems*, In: Proceedings of the Workshop on Personalization in eCommerce at the Second International Conference on Adaptive Hypermedia and Web-Based Systems (AH-02), Universidad de Malaga, Spain, 80-89.

McGinty, L. & Smyth, B. (2002a) *Comparison-Based Recommendation*, Lecture Notes of Computer Science 2416, Proceedings of the 6th European Advances in Case-Based Reasoning, Springer, 575-589.

Mittal, S. & Falkenhainer, B. (1990) *Dynamic Constraint Satisfaction Problems*. Proceedings of 8th National Conference on Artificial Intelligence, AAAI-90, 25-32.

Mittal, S. & Frayman, F. (1989) *Towards a Generic Model of Configuration Tasks*. In: Proceedings of International Joint Conference on Artificial Intelligence, IJCAI-89, 1395-1401.

Pretschner, A. (2001) *Classical search strategies for test case generation with Constraint Logical Programming*. In: Proceedings of Workshop on Formal Approaches to Testing of Software, Aalborg, Denmark, 47-60.

Pu, P., Faltings, B., & Torrens, M. (2003) *User-Involved Preference Elicitation*, In Proceedings 18th International Joint Conference on Artificial Intelligence (IJCAI'03), Workshop on Configuration, Acapulco.

Reiter, R. (1987) *A theory of diagnosis from first principles*. Artificial Intelligence 23(1), 57–95. Preece, A., Talbot, S., Vignollet, L. (1997) *Evaluation of Verification Tools for Knowledge-Based Systems*, International Journal of Human-Computer Studies, 47, 1997, 629-658.

Rochart, G., Jussien, N., & Laburthe, F. (2003) *Challenging explanations for global constraints*, in: CP'03 Workshop on User-Interaction in Constraint Satisfaction, Ireland.

Royalty, J., Holland, R., Goldsmith, J., & Dekhtyar, A. (2002) *POET: The online Preference Elicitation Tool*, In: Proceedings of AAAI'02 Workshop on Preferences in AI and CP: Symbolic Approaches, Edmonton, CA.

Rumbaugh J., Jacobson I., & Booch G. (1989) *The Unified Modeling Language Reference Manual*. Addison-Wesley.

Schmitz V., Leukel J. & Kelkar O.: (2004) *XML-based Data Exchange Of Product Model Data in E-Procurement And E-Sales: The Case of BMECAT 2.0*, In: International Conference on Economic, Technical and Organisational aspects of Product Configuration Systems, Kopenhagen, Denmark, 97-108.

Stumptner, M. & Wotawa, F. (1999) *Reconfiguration using Model-based Diagnosis*, In: Proceedings of the International Workshop on Diagnosis (DX99), Scotland.

Sung, H.H. (2002) Helping Customers Decide through Web Personalization, IEEE Intelligent Systems, 17(6), 34-43.

Thompson, C.A., Goeker M.H., & Langley, P. (2004) *A Personalized System for Conversational Recommendations*, Journal of Artificial Intelligence Research, 21, 393-428.

Tsang, E. (1993) *Foundations of Constraint Satisfaction*, Academic Press, New York.

van Harmelen F., Patel-Schneider P.F., & Horrocks I. (2001) *A Model-Theoretic Semantics for DAML+OIL*. Retrieved in Dezember 2005 from http://www.daml.org.

Warmer J., & Kleppe A. (2003) *The Object Constraint Language 2.0*. Addison Wesley.

Yokoo M., Durfee E.H., Ishida T., & Kuwabara K. (1998) *The distributed constraint satisfaction problem*. IEEE Transactions on Knowledge and Data Engineering, 10(5), 673-685.

Yokoo, M. (2001) *Distributed Constraint Satisfaction*. Springer Berlin New York.

Zanker, M. (2002) *Distributed Configuration*, PhD thesis, University Klagenfurt, Austria.