WEB-BASED CONFIGURATION OF VIRTUAL PRIVATE NETWORKS WITH MULTIPLE SUPPLIERS

A. FELFERNIG, G. FRIEDRICH, D. JANNACH, AND M. ZANKER
Computer Science and Manufacturing Research Group
University of Klagenfurt, Austria
{felfernig, friedrich, jannach, zanker}@ifit.uni-klu.ac.at

Abstract. An Internet-based Virtual Private Network (IP-VPN) uses the open, distributed infrastructure of the Internet to transmit data between corporate sites. The configuration (network design) for a specific customer network typically requires the selection of network access lines and backbone sections that are provisioned by different organizations in a supply chain. Moreover, when configuring such a network, the given customer requirements (e.g., minimal bandwidth) have to be observed.
Within this paper, we show how the (sales-)configuration process for these networks is supported within the CAWICOMS[1] framework for distributed configuration. Beside the implementation of an adequate distributed problem solving mechanism based on Constraint Satisfaction with commercial tools, we address the problem of supplier selection and knowledge integration in a Web-based environment for eCommerce: Based on common ontological commitments on representation concepts for the configuration domain, the suppliers can advertise their products and services, whereby the distributed problem solving process involves locating and executing the supplier's configuration service using an open XML-based protocol.
We present the architecture of the implemented prototype framework and show the relation of our work to emerging approaches in the fields of Distributed Problem Solving and Semantic Web Services.

[1] CAWICOMS is the acronym for *Customer-adaptive Web interface for the configuration of products and services with multiple suppliers*. This work was partly funded by the EU through the IST Programme under contract IST-1999-10688. (www.cawicoms.org)

1. Introduction

In today's networked economy, effective communication has become a necessity, when remote users (e.g., sales people or distant offices) need easy access to the corporate network and secure connections with the business partners are required. Virtual Private Networks (VPN) extend the company's intranet and are capable of providing such services at reduced cost using the worldwide IP network services and dedicated service provider IP backbones (Infonetics Research, 1997). VPN infrastructures are designed to be flexible and configurable in order to be able to cope with a rich variety of possible costumer requirements. Therefore, the establishment of a concrete VPN involves different steps after determination of customer requirements like locations to be connected or specification of required bandwidth: selection of adequate access facilities from the customer site to some entry point to the VPN backbone, reservation of bandwidth within the backbone, as well as configuration of routing hardware and additional *services* like installation support. Note, that it is very unlikely that all these products and services needed for the implementation of the VPN can be supplied by one single organization but are in general made available by specialized solution providers, e.g., Internet Service Providers, telecommunication companies or hardware manufacturers (see Figure 1). Therefore, VPNs are typically marketed by specialized resellers (or telecommunication companies like two of our application partners) that integrate the services of individual suppliers and offer complete VPN solutions to their customers.

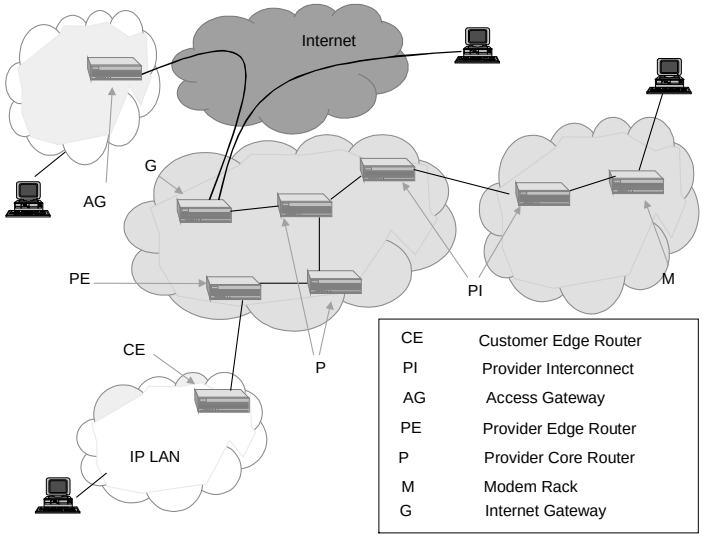

Figure 1. IP-VPN sketch

Efficient sales processes for such complex and configurable products and services require specialized support by sales-force automation tools like product configurators with advanced problem-solving capabilities (Fleischanderl et al, 1998; Mailharro, 1998). Consequently, product configuration has become an important application area for Artificial Intelligence-based techniques in industry and nearly all vendors of Enterprise Resource Planning or Business-to-Business eCommerce systems have integrated such technologies in their products (Haag, 1998; Yu and Skovgaard, 1998). However, while there are several commercial product configuration tools on the market, there are some specific requirements in the described application domain that are not addressed adequately by nowadays systems:

- *Distributed configuration*: due to the permanent physical restructuring of the network infrastructure, appearing and disappearing suppliers, and corporate privacy reasons, an approach with one single centralized knowledge base and problem solver is not possible. Furthermore, means must be provided in order to locate the appropriate suppliers and initiate the provided configuration service.
- *Heterogeneity*: For the detailed configuration of the network, the suppliers may employ some specialized software performing complex computational methods for network routing or are using legacy configuration systems. These systems must interact in order to cooperatively solve the overall configuration task.
- *Knowledge integration*: Resellers and network suppliers may use different concepts and terminology for the configuration task. Therefore a common ontology as well as a knowledge exchange mechanism for the application domain is needed.

Exactly these issues are addressed as part of the EU-funded CAWICOMS project and will be discussed in the rest of the paper: First, we describe how standard configuration technology can be extended to cope with Distributed Problem Solving requirements and sketch the implementation framework developed within CAWICOMS. Later on, we discuss how web-based eCommerce between the involved suppliers is done based on information-providing (semantic) web services (McIlraith et al, 2001).

2. Configuration of Virtual Private Networks

At a first glance, the problem of finding routes through a network that connect several given access points and observe specific constraints (e.g., on bandwidth) does not fit to the widely adopted component-connection oriented definition of product configuration from (Mittal and Frayman, 1989) which is the basis for most configuration tools.

However, the sales process for VPNs comprises several stages: First, a high-level, abstract design of the VPN is performed, i.e., for each of the customer sites to be connected we have to select a way to connect the site to some entry point in the IP-backbone network. Typically there are several choices, e.g., on the type of the backbone access (which entry point, which supplier, required protocols etc.), and there are specific constraints (e.g., on compatibility of protocols) that have to be observed. In addition, we have to assure that there is a route within the backbone that connects all the chosen entry points.

The result of this first phase is a coarse network layout and a *price* which is used to generate an offer for the customer. Once this offer is accepted, the detailed configuration of the VPN can be performed, i.e., computation of low-level technical details like IP-addresses or router configuration parameters. Note that this step requires the usage of (existing) specialized routing or configuration software which is for instance capable of taking into account the current network load of some supplier network.

The following simplified example (Figure 2) will illustrate the rationale of hierarchical configuration of VPNs. The sales engineer for VPNs interacts with the configuration system and interactively enters the customer requirements: The customer sites in *London*, *Paris*, and *Milan* have to be connected, whereby for each connection, he can enter requirements on e.g., bandwidth or latency.

In a first step, the configurator determines a set of adequate access lines from all the available lines that were advertised by the suppliers. Furthermore, a route in the backbone network is computed that interconnects the selected access lines. Note, that this computation is done with the standard functionalities of our configuration software, which also allows for optimization. Optimization can be done according to some objective function like *price* or number of needed suppliers. Moreover, the search can be guided by user preferences (e.g., according to some business goals) (Junker, 2001).

This search process results in the high-level network marked with bold lines in Figure 2 and determines the set of needed suppliers (*BTT, TELCO Paris*). In a next step the configurators of these suppliers are contacted in parallel, whereby information about the required components of the network is handed over in the format defined for knowledge sharing. At the supplier sites, details of the network layout are computed (or simply read from a catalog), whereby this typically involves specialized, existing software modules. Those parts of these computations that are relevant for the reseller are transformed back to the common format (according to the ontology) and returned to the main configurator.

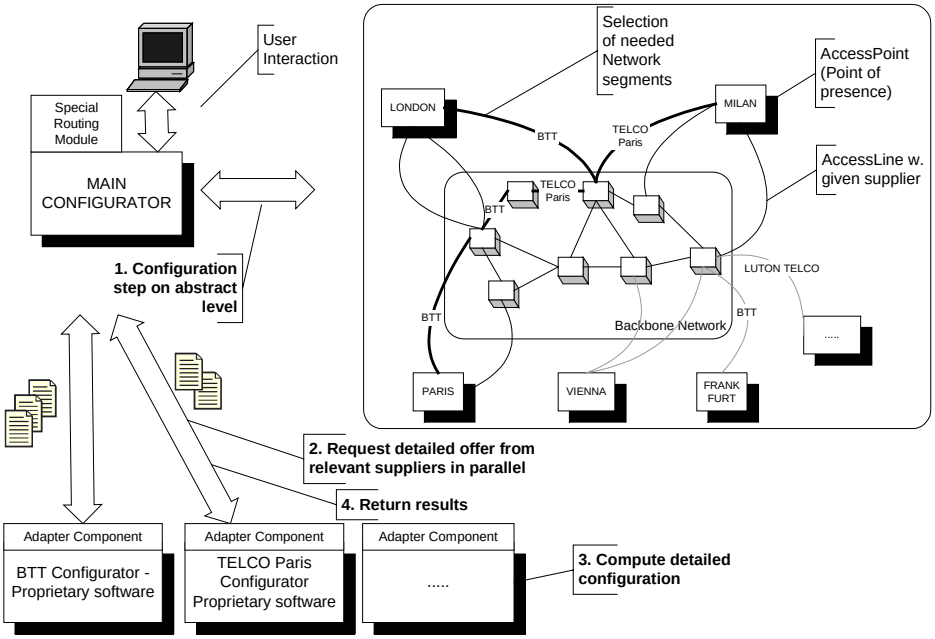

Figure 2. Example scenario

3. Knowledge sharing

In order to allow communication among several systems, the CAWICOMS framework must be able to support these given business processes and implement that sort of *hierarchical configuration process*, whereby the integration of existing specialized software modules is a crucial factor. Figure 3 illustrates the overall rationale of knowledge sharing in our framework.

In a first step, the participating companies have to agree on a shared ontology (and terminology) for the domain of IP-VPNs. While this problem is far from being solved in the general case, it was already shown that for specific application domains (e.g., *RosettaNet* for Electronic Components; (RosettaNet, 2001) such standardization is possible. Moreover, in our application domain the companies involved in the supply chain typically rely on long-term business relations and negotiation phases, which alleviates these integration steps.

The integration of the different product models is supported in the CAWICOMS framework as follows: We use UML as a domain-independent graphical modeling language for the design of the common product model (Felfernig et al, 2001; Rumbaugh et al, 1998). This method has the advantage of being in wide-spread use and comprehensible for domain

experts and is expressive enough for the configuration domain. Moreover, this representation mechanism is independent from the proprietary notation of specific configuration tools. Finally, the models acquired in UML (*Unified Modeling Language*) can be automatically transformed both into the representation of the configuration engine, in our case ILOGs JConfigurator; (Ilog, 2001) as well as into other ontology description mechanisms like DAML+OIL (Fensel et al, 2001). Figure 4 shows a fragment of the VPN product model implemented in the current prototype:

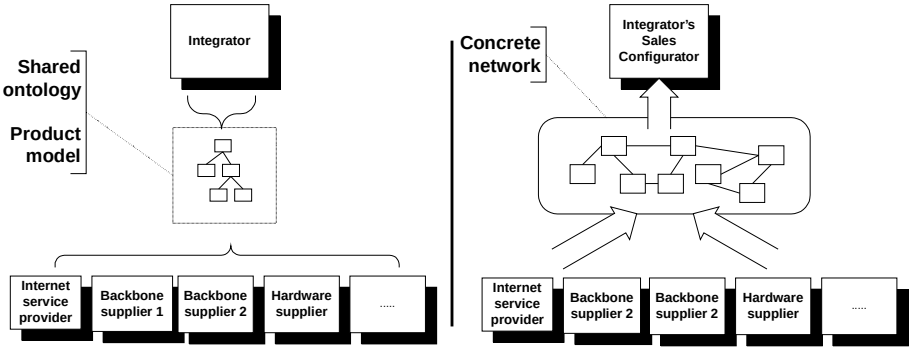

Figure 3. Shared ontology & concrete networks and catalogs

In general, an IP-VPN consists of a set of *AcccessPoints* which are connected to *BackboneSections* via *AccessLines* whereby these classes have configurable attributes. The inner IP-backbone network is given by interconnections between the backbone sections (*ProviderInterconnect*). Furthermore, additional routing hardware etc. will be part of the generic product model. This model represents the common ontology for all the companies that are involved in the supply chain.

Once a common understanding of the problem domain is established between the involved parties, knowledge about concrete offers by the suppliers has to be exchanged and integrated. Within our framework, this is done by explicit *advertisement* of specific instances or subtypes of ontological concepts: As an example, think of a supplier that offers some specific means of access to some backbone section. In order to do that, it publishes the available service to the configuration system of the *integrator* including some concrete values (e.g., *I can provide a managed firewall connection from London to BackboneSection b1 with guaranteed bandwidth of 500kb/s at price X etc.*). Note that this advertisement has to be done automatically by registering the offer to an integration agent at the *reseller's* site, because of changes in the range of products product and available resources at the supplier. These offers then form the concrete available network (as well as other services like installation support) for the integrator, which is depicted on the right hand side of Figure 3. As a

knowledge representation format for these offerings, we can utilize an XML representation for UML (instance) diagrams. The usage of the emerging DAML+OIL standard is currently evaluated and will be included in future implementations. Note however, that the supplier systems, which can be fully-fledged product configurators, specialized routing software modules, or mere product catalogs, do not have to rely on the same internal knowledge representation or problem solving mechanism like the configurator at the reseller's site. The only requirement at that stage is that the published portions of the e.g., catalog information can be transformed to the common ontology. Typically, the computation of details on the network at the supplier site will include additional reasoning mechanisms or data from other sources like current network load which is not relevant for the integrator in the first place.

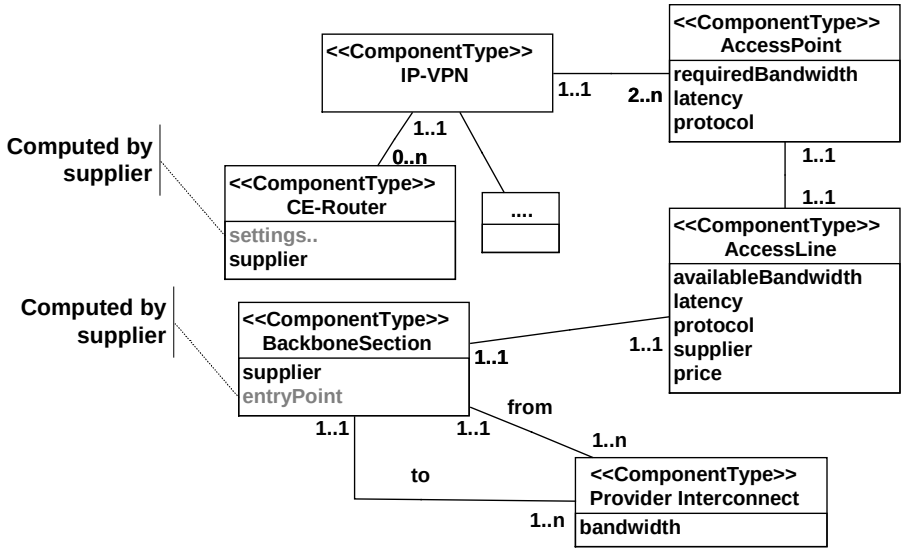

Figure 4. Product model fragment for VPNs in UML

4. Distributed Configuration

In recent years, some major advances have been made in the field of Distributed Artificial Intelligence and in particular with respect to techniques for Distributed Constraint Satisfaction (Yokoo, 2001) and Multi-Agent Systems. In addition, the field is continuously pushed forward by the rapid growth of world-wide Business-to-Business eCommerce and the need for seamless supply-chain integration. Having discussed the problem of knowledge integration and –exchange in the previous section, we will now

focus on the techniques and algorithms employed for distributed problem solving in the CAWICOMS framework with respect to specific requirements for our application domain. We have to state in advance that the general conditions of our domain-independent framework for distributed configuration include both (re-)use of existing technologies as well as openness for integration of other systems[2]. Therefore, two quite different application domains where chosen for evaluation, whereby one addresses configuration of VPNs as described here and the other the more traditional product configuration of telecommunication switches.

We base our algorithms on Constraint Satisfaction techniques, which have shown to be adequate for solving configuration problems with regard to expressiveness, efficient problem solving, and declarative knowledge representation (Fleischanderl et al, 1998; Mailharro, 1998). While there are already algorithms available for Distributed Constraint Satisfaction (Silaghi et al, 2000; Yokoo, 2001), these approaches have some characteristics that do not fit into our framework too well:

- they require the usage of specialized search mechanisms like Asynchronous Backtracking or Weak-Commitment Search not provided by commercial tools.
- in many cases, the problem is simplified to agents that only handle binary constraints and one single local variable (although, in general, these approaches can be extended).
- they do not take the given supply chain structure and existing business processes into account for (configuration) problem solving.

The approach taken in the CAWICOMS framework takes these considerations into account by allowing the integration of several configurators in the supply chain, whereby – compared with other approaches – there is some predefined, typically tree-structured order and a client-server relation between the involved systems; at the inner nodes of the tree-structured supply chain setting, dedicated configurators facilitate the integration of serving configuration systems. At the top of the supply chain a *main configurator* communicates with the user via an interaction module.

The main idea of integrating the involved systems relies on *sharing of variables* in terms of Constraint Satisfaction and *sharing of components* in the sense of (Mittal and Frayman, 1989). Configurators may share parts of their configuration knowledge with neighboring systems; more specifically, configurators may *publish* relevant parts of their knowledge to others that are at the next higher level in the supply chain. This knowledge is then incorporated (including the definition of inter-agent constraints) into the product model at the next higher level and we markup those "imported"

[2] For an overview of the complete project, please refer to (Ardissono et al, 2001).

chunks of knowledge at the higher level with information, which supplier is *responsible* for finding a solution for that sub-problem. During the configuration process, search starts at the top level of the supply chain, e.g., using an extended standard CSP search algorithm like implemented ILOG JConfigurator (Ilog, 2001); during the search for solutions for the *local* configuration problem at one node we may encounter a sub-problem which references to a suppliers knowledge base.

Figure 4 depicts this situation, where some of the attributes of the integrated product model are shaded, i.e., we know that the values for these variables have to be determined by some supplier configuration systems. In the example, these are technical details that cannot/should not be computed by the reseller's configurator, because the rationale of the computation is both complex (needs specific algorithms) and confidential to the supplier.

Obviously, it is not sufficient that each of the involved configurators finds a solution to its local sub-problem but we have to find an overall solution that satisfies both the user requirements as well as all the given constraints between the involved configurators. In (Felfernig et al, 2001a), the overall conditions for finding a globally consistent solution to such a distributed problem are described in terms of a logical model of distributed configuration. In general, means of conflict resolution have to be provided in cases, where a solution that is consistent with the local configuration knowledge is inconsistent with other partial solutions of configurators that share some variables. As an example, in (Felfernig et al, 2001b) we describe such a sound and complete algorithm for distributed configuration of telecommunication switches, which is based on synchronous backtracking to ensure global consistency. This algorithm was implemented by extending to the standard forward-checking backtracking search procedure of ILOG JConfigurator and takes the mostly sequential nature of the task into account[3].

In principle we can use that algorithm also for the configuration of VPNs, according to our intention of being domain-independent with respect to our solving techniques: we start selecting lines in the abstract network using the configurator at the reseller's site, request a detailization of the solution from the supplier, integrate the results and continue with the next access line. In case of inconsistencies that may arise during integration of the detailed results we backtrack and search for another solution.

In the case of VPN configuration, however, we can apply a hierarchical and parallel approach, where we compute the complete (optimal) solution at the abstract level and then request the configuration details from all the involved suppliers in parallel. This can be done under the assumption that

[3] Note, that *parallel* computation (like it is in Distributed CSPs) is not a natural way of problem solving in some domains.

the suppliers will always find a solution given the parameters for their sub-problem and that the returned results do not violate any constraints when integrated into the overall solution, which is a reasonable assumption according to the requirements of our application partners.
A further requirement for the application domain can be tackled with that approach: When requesting a solution (or an offer) from a supplier it may be the case that the configuration process at the supplier side requires some (longer-lasting) human interaction because e.g., some internal business processes have to be initiated. This implies that the overall configuration can only be completed after *all* the results from all involved suppliers are returned. Consequently, after sending out the requests for solutions to the suppliers in parallel, we have to wait until the last solution is replied before notifying for instance the sales representative that the configuration is completed. This requirement is addressed in our framework by supporting long-lasting configuration sessions. In some cases, however, it may also be sufficient to present the user the results of the high-level network for the offering phase.

5. Implementation

The CAWICOMS framework for distributed configuration is implemented on Sun's JAVA-based J2EE platform that supports component-oriented development and portability and provides basic standard functionalities for Internet-based programming like naming services or load balancing. *Java Server Pages* are used for the generation of adaptive interfaces which is part of the project but out of the scope of this paper.

Reasoning. The core reasoning mechanism for distributed configuration is implemented by extending the commercial domain-independent configuration engine *JConfigurator* from ILOG[4]. Note however, that these extensions were done without changing the core mechanisms of the configurator engine but only by using the built-in extensibility features. This was done in order to keep the solution as independent as possible from specific vendors. The main requirement for a configurator to be usable in our framework is that one can perform user-defined procedures at certain points (e.g., when trying to solve a goal) in the search process. So whenever the configurator (or simply a constraint solver) tries to configure a certain component instance, one module of our framework checks whether the component has to be configured by a supplier by querying a database. If so, we have to look up the supplier and request a solution for the sub-problem and integrate the results in the local solution space. Finally, we added some

[4] www.ilog.com

generic interface to plug-in specialized domain-specific algorithms for finding a route within the backbone network, in our case a variant of the *minimum spanning tree* algorithm.

Knowledge Acquisition & Representation. The CAWICOMS workbench includes a *Knowledge Acquisition Workbench* which must be general enough to model a wide range of configuration problems. As already described we made excellent experiences by using UML (and the built-in *Object Constraint Language*) as a language for expressing configuration problems on a conceptual level and we have shown that it can also be used to model problems like in the VPN domain that do not fit the classical scheme of component-connection oriented configuration in the first place. In the current state of the project, we take the XMI representation of UML (which can be generated by most commercial UML tools like *Rational Rose*) and transform it automatically into an intermediate XML representation which is then used to incorporate additional knowledge both about potential suppliers as well as personalization information for the user-interface generation task. Again, depending on the specific configuration tool that is used for the application domain, we have to write adapter components to transform the knowledge into the representation of a specific tool (Figure 5). Future steps include the usage of DAML+OIL (Fensel et al., 2001) as knowledge representation format.

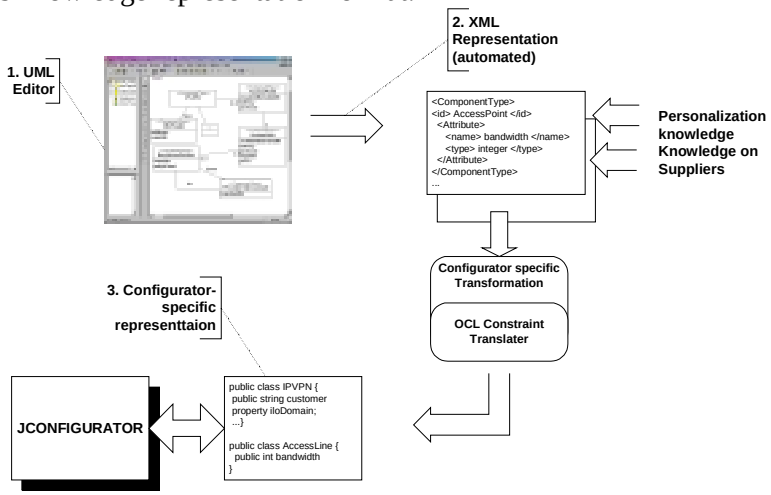

Figure 5 Schematic knowledge acquisition process

Knowledge exchange during the configuration process. During the distributed configuration process, the configuration agents have to exchange information about the current state of the search process. On the one hand, when requesting a solution for a sub-problem from a supplier, the requesting agent has to inform the supplier both of the actual requirements as well as of the intermediate results of the inference process (e.g., domain reductions in

CSP terminology). On the other hand, the results computed by the supplier configurators have to be returned to their clients. Note, that in the configuration domain these results will be complex data structures that reflect the computed configuration including instantiated components, attribute values (or domains) as well as connections between the individual components. In UML terminology, we have to exchange *instance models* (containing individual objects rather than classes) whereby these instance models have to conform the structure defined in the product model, i.e., the static structure diagram like in Figure 4. Finally, these pieces of information also have to be communicated via the User Interface component, through which customer requirements can be stated and results will be retrieved and presented. From a low-level technical point of view on communication in the Internet, there are several techniques available that allow communication between processes over the network. The most prominent ones are CORBA, see e.g., (Mowbray and Ruh, 1998), Microsoft's DCOM, or Java-based Remote Method Invocation (RMI). These approaches allow the exchange of complex data structures and CORBA and DCOM allow for interoperability between different programming languages. However, these techniques are still *programming* approaches that offer remote computation but do not cope with the requirements of offering *semantic services*, that can be located and accessed on basis of their functionality. Moreover, these techniques are often rather complex to use and rely on low-level TCP/IP communication which may contravene a company's security policy.

With the development of SOAP (Simple Object Access Protocol – see http://www.w3c.org) these limitations are overcome by defining a XML-based protocol for information exchange over the Internet. This protocol aims at providing means for describing what is *in* a message and how to process it, conventions for remote procedure calls as well as encoding rules for application-defined data types. The goals therefore include the possibility to add *semantics* to messages as well as a platform-independent communication mechanism over HTTP.

The approach taken in the CAWICOMS framework relies on the same mechanisms, whereby some specific extensions for the domain are incorporated (See Figure 6). Communication between configurators as well as with the user interface application is based on *ILOGs WebConnector* protocol. This protocol (and toolkit) was basically designed as a generic protocol to publish and edit complex data structures over the Web, and is not limited to the domain of product configuration. It defines an API to manipulate complex data structures whereby the calls to the API can be done in a SOAP-compliant XML format and the results are again returned in XML (i.e., XML-Schema) which can be further transformed for e.g., presentation to the user. ILOGs JConfigurator was integrated with the

WebConnector toolkit, which required the definition of how manipulations to the data structures on the abstract layer (visible for the clients) are mapped to the internal object model of the configurator. Thus, the configurator can be accessed via the WebConnector protocol using XML messages and all the transformations from and to the XML format are performed automatically. In our setting of distributed configuration we use this protocol for communication not only with the user interface but also for communication between individual configuration systems.

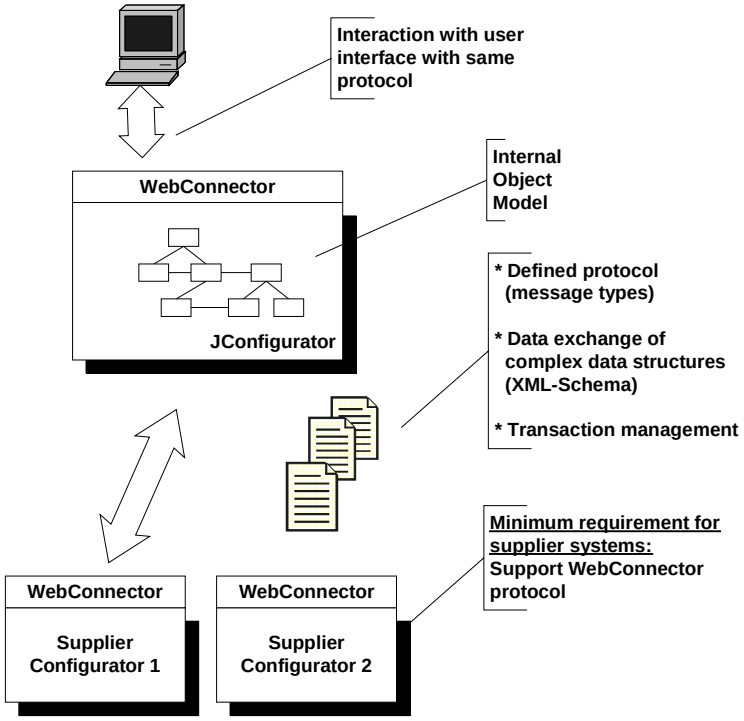

Figure 6 XML-based information exchange

The built-in extensibility features of the WebConnector toolkit were used to add product configuration–specific methods and transaction management. Finally, a module was developed that automatically adapts the contents of the messages according to the *views* on the product model of the involved configurators: This is done because according to our notion of *sharing* knowledge, the cooperating configurators have only restricted knowledge of the product model of each other.

Another design goal that should be reached with this approach is openness to different configuration tools or legacy systems. As depicted in Figure 6, supplier configurators (or existing systems) do only have to support the open WebConnector protocol in order to participate in the distributed (network) configuration process. However, in these cases,

adapter components have to be developed that map the XML-messages onto the internal representations of these systems.

6. Distributed configuration (of VPNs) as Semantic Web Service

The enormous growth of the Internet and related businesses that operate on top of the Web causes an increasing demand for attaching meta-data to the information and services provided, once we do not solely view the Web as a large repository for text and images (McIlraith et al., 2001). Such annotations – that have to exceed the descriptive capabilities of simple keywords – aim at describing the *semantics* of the pieces of information available on the Web, thus giving us the possibility to perform more sophisticated operations, i.e., high-level queries on data as well as identification, location and execution of available services. Especially for the case of services that are provided online (like e.g., flight reservation or travel planning) we face the problem that these services are only accessible via custom-made Web interfaces and the usage of the services is limited to humans that interact with the systems. Moreover, once one has solved the problem of finding such a service, in general, no standard way of interacting (protocol) or a common terminology is available for these services.

However, with the rapid emergence of Business-2-Business (B2B) eCommerce and electronic market places, the automation of the interaction between the involved systems has become inevitable. This automation involves both the definition of a standard *ontology* containing the concepts of the business domain as well as the definition of *services* that are to be provided in the domain. In recent years, a lot of XML-based (pseudo-) standards like Commerce XML (cXML) or Common Business Library (CBL) for B2B applications have evolved that support communication on the basis of standardized terminologies, data exchange formats and a set of predefined operations like e.g., order placement or other typical business transactions. So while these standard operations are quite well understood and supported in today's systems, i.e., the meaning of an *purchase order* is basically the same for most businesses, there are no mechanisms available to describe the semantics of non-standard and domain-specific world-altering operations like reservation of an airline ticket like described in (McIlraith et al., 2001). The same holds for the domain of configurable products where several interaction steps like the definition of requirements or search for a suitable configuration may be required. This problem is even harder in cases where configuration of products may occur on several occasions in a supply

chain and different (configuration) services are required to compute a satisfying solution.

The further development of the CAWICOMS framework aims at providing distributed configuration services in supply chain settings based on an open Semantic Web environment. While much of the work done in the emerging Semantic Web community is still based on theoretical considerations, the application domain of distributed product and service configuration can serve as a test bed for adding intelligence to the Web: Firstly, there is real industrial demand and business opportunities in this domain, and secondly, the domain is sufficiently restricted and understood in order to allow the implementation of viable solutions. However, the transformation of the distributed configuration process into a Web Service in the sense of (McIlraith et al., 2001) requires more than just the adoption of emerging standards on the technical level, like the usage of SOAP-conforming messages and DAML+OIL for knowledge representation or techniques for service localization. It rather requires a common understanding of

a) the configuration problem itself, i.e., the semantics of the concepts used to describe configuration problems,
b) the services required to solve a configuration problem as well as the semantics and consequences of service execution,
c) the means of exchanging information during the configuration process, i.e., how do we represent complex data structures and what is the meaning of individual pieces of information.

In recent years, both the academic (Mittal and Frayman, 1989; Peltonen et al, 1998) and the industrial communities (Fleischanderl et al., 1998; Junker, 2001; Mailharro, 1998) involved in the product configuration domain have made significant advances in establishing a common component-connection oriented view of the configuration task. The theoretical foundations in the CAWICOMS project are based on a logic theory of the (distributed) configuration task. This allows both for precise semantics for the employed concepts as well as independence from specific knowledge representation and reasoning mechanisms[5]. In general, a configuration problem consists of a *domain description* that describes the available component types, their attributes and connection points and some specific user requirements for the actual configuration task. The configuration result can be described by grounded literals, whereby we define the set of predicate symbols that describe a configuration result as *CONL*. (see Felfernig et al., 2000).

[5] Note, that this does not require the involved configurators to rely on such a representation mechanism.

Definition (Configuration problem): *A configuration problem is described by a triple (DD, SRS, CONL), where DD and SRS are sets of logical sentences and CONL is a set of predicate symbols. DD represents the domain description, SRS the system requirements specification for a configuration problem instance. A configuration CONF is described by a set of positive ground literals whose predicate symbols are in CONL.* ❑

Based on this definition, we can now describe in which situation *CONF* is a solution for the configuration problem:

Definition (Consistent configuration): *Given a configuration problem (DD, SRS, CONL), a configuration CONF is consistent iff* $DD \cup SRS \cup CONF$ *is satisfiable,* ❑

To ensure the completeness of a configuration, additional formulae for each symbol in *CONL* have to be introduced to *CONF*. We denote the configuration *CONF* extended by these axioms with $\overline{CONF}$.).

Definition (Valid configuration): *Let (DD,SRS,CONL) be a configuration problem. A configuration CONF is valid iff* $DD \cup SRS \cup \overline{CONF}$ *is satisfiable.* ❑

In (Felfernig et al., 2001a) this logical theory was extended to the case where several configuration agents solve a distributed configuration problem, whereby the individual agents solve sub-problems of the overall problem and rely on local knowledge bases (domain descriptions).

Definition (Distributed configuration problem)*: A distributed configuration problem for n different configuration agents is described by a triple* (DD_{set}, SRS_{set}, *CONL) where*
$DD_{set} = \{DD_1 \ldots DD_n\}$ *and*
$SRS_{set} = \{SRS_1 \ldots SRS_n\}$.
Each element of DD_{set} *and of* SRS_{set} *is a set of logical sentences and CONL is a set of predicate symbols. For* $k \in \{1 \ldots n\}$, DD_k *corresponds to the domain description of the configuration system k and* SRS_k *specifies its system requirements. A configuration CONF is described by a set of positive ground literals whose predicate symbols are in CONL.*❑

Definition (Valid solution to a distributed configuration problem)*: Given a distributed configuration problem(*DD_{set}, SRS_{set}, *CONL), a configuration CONF is valid iff* $DD_k \cup SRS_k \cup \overline{CONF}$ *is satisfiable* $\forall k \in \{1 \ldots n\}$ ❑

In principle, the involved configurators can be seen as independent modules that are able to solve their individual configuration tasks. We define a property called *defined interfacing*, where all involved configuration systems employ disjoint sets of predicate symbols except for those contained in CONL. This way configurators can exchange (partial) configuration results based on shared predicate symbols in *CONL*. Based on

this property, we can define precisely under which circumstances the distributed solving of the configuration task generates equivalent solutions to a centralized approach.

> ***Theorem:*** *Let (DD, SRS, CONL) be a configuration problem and (DD_{set}, SRS_{set}, CONL) a distributed configuration problem with defined interfacing where* $DD = \bigcup_{dd \in DDset} dd$ *and* $SRS = \bigcup_{srs \in SRSset} srs$.
>
> *CONF is a valid configuration for (DD, SRS, CONL) iff CONF is a valid solution for the distributed configuration problem (DD_{set}. SRS_{set}, CONL).* ❑

For the **Proof** and further details, see (Felfernig et al., 2001a).
Finally, when solving the distributed configuration problem, agents exchange partial solutions to come to an overall configuration. During the search process it could be discovered that some of the exchanged partial solutions are in conflict with the local knowledge base and conflict resolution among agents must be initiated.
This general definition of the (distributed) configuration task serves two purposes in our framework. Firstly, we can extend the formalism by introducing the extensible set of predicate symbols *CONL* for component-connection oriented configuration, i.e., *CONL={type/2*, *conn/4*, *val/3}* for describing component instances, connections and attribute valuations as the common interface for exchanging (partial) configurations. Other representations (e.g., those of commercial tools) can be mapped to that logical representation. Secondly, if we standardize the way the components (and their attributes) itself are to be described in *DD*, we can automatically transform several other representation mechanisms for product models (like *UML*) into the logical framework thus yielding precise semantics for the employed modeling concepts (see Felfernig et al., 2000). In addition, the logical framework does not explicitly require some specific reasoning mechanism because it rather describes the conditions under which a configuration instance is a valid solution for a distributed problem. The reasoning task can e.g., be accomplished – as in the CAWICOMS framework – by some specialized constraint solver.

Finally, we need to address the description of the actual service a configurator can offer to its clients in a distributed environment, whereby two different pieces of information have to be provided:

1. *which products* can be configured by the configurator?
2. *what capabilities* (services) does the configurator offer?

The first point is related to the advertisement of the range of products the configurator is capable to configure. While there are approaches emerging to define a unique world-wide catalog and categorization of products like UNSPSC or eCl@ss in B2B eCommerce (see, e.g., Fensel et al, 2001b), up to now there is no agreed-upon industrial standard. Moreover, most of these classification schemes do not support products that are parametrizable which is typical for configurable artifacts. The approach taken in CAWICOMS, which relies on shared ontological commitments for configuration domain specific representation concepts and advertisement of services, was already described in Section 3. Future work in CAWICOMS will therefore include an approach to integrate these techniques with forthcoming classification standards. However, the way of ontology and knowledge integration between the involved partners strongly depends on the type of businesses: For short-term businesses of standardized (low-priced) products (like purchasing an airline ticket, see McIlraith et al., 2001), the usage of world-wide classification standards will be a must in order to allow the participation of a large number of possible providers of such products. In contrast, in highly complex domains like the provision of Virtual Private Networks, supply chain integration will typically rely on longer lasting business relations among the partners; therefore, we suppose that ontologies for specific application domains or even only for the involved partners can be established. Nonetheless, better techniques and tools for ontology construction and integration will be needed and within the scope of future development of CAWICOMS.

Another point targeting *service* description relates to the individual capabilities of the involved configuration systems. Note, that we do not want to restrict our framework to some specific tool or specific constraint solving algorithms, because openness towards legacy configurators is an important prerequisite. In the framework described in (Felfernig et al., 2001a) we base distributed problem solving on exchanging partial configurations and conflicts. Furthermore, we define basic communicational capabilities a configurator has to provide in order to participate in a distributed configuration process. Note, that the very semantics of these *method calls* have to be described precisely in order to allow successful cooperation. Therefore, we rely on a logic theory of configuration, that describes the semantics of the required services unambiguously. For a simple distributed algorithm using a facilitating agent (Felfernig et al., 2001a) we need some basic message types exchanged between configuration agents, e.g.:

- *requesting a solution*: As an input, a partial configuration (containing the user requirements) is given. The return value of the call is either a configuration that is consistent with the local knowledge base or else a notification if no solution can be found and ideally the conflict.
- *adding a conflict*: In order to avoid infinite processing loops, conflicts are exchanged among agents and incorporated into their view of the overall problem solving status.

Furthermore, there will be some additional methods for agent initialization as well as additional parameters to control the search process, since – as opposed to simple web services where the individual calls to some agent will be independent – the distributed configuration process will involve a series of subsequent calls to an agent (e.g., in case of backtracking) in a configuration session.

7. Conclusions

We have presented a framework for distributed (sales) configuration and subsequent offer generation that is currently being developed in the EU-funded CAWICOMS project. While the general framework is designed to be applicable to many different domains, we have focused on the configuration of Virtual Private Networks because of its specific requirements on the configuration process and its industrial importance.

After describing the specific properties of the application domain, we sketched an approach for knowledge sharing among configurators in a supply chain based on a common ontological commitments on concepts for problem representation and advertisement of configuration capabilities. After discussing the requirements for distributed and cooperative problem solving in a web-based environment, we presented the current implementation of the CAWICOMS framework that is built using state-of-the art component technology. Distributed configuration is performed by extending an industrial-strength constraint solver for the problem solving process and by using an open, XML-based knowledge exchange mechanism.

In the final section, the relation of the problem setting to the emerging field of the *Semantic Web* is emphasized. Based on a formal definition of the distributed configuration problem with precise semantics, we outlined the future extension of the framework in order allow the provision of configuration s*ervices* in an open web-based environment.

References

Ardissono, L., Felfernig A., Friedrich G., Jannach D., Schaefer R., and Zanker M.: 2001, Intelligent Interfaces for Distributed Web-based Product and Service Configuration. *Proc. Web Intelligence (WI-2001)*, Maebashi, Japan, *Lecture Notes in Artificial Intelligence*, Springer Verlag.

Felfernig A., Friedrich G., and Jannach D.: 2000, UML as domain specific language for the construction of knowledge-based configuration systems, *International Journal of Software Engineering and Knowledge Engineering*, Vol.10(4), pp. 449-470.

Felfernig A., Friedrich G., and Jannach D., and Zanker, M.: 2001a, Towards Distributed Configuration. *Proc. KI-2001, Joint German/Austrian Conference on AI*, Vienna, Austria, Lecture Notes in AI, Springer Verlag.

Felfernig A., Friedrich G., and Jannach D., Russ, C., and Zanker, M.: 2001b, Multi-site product configuration of telecommunication switches, *20th IASTED Conference on Applied Informatics*, Innsbruck, Austria, 2002.

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

Fensel D., Ding, Y., Omelayenko, B., Schulten, E., Botquin, G., Brown, M, a. Flett, A.: 2001, Product Data Integration in B2B E-Commerce, *IEEE Intelligent Systems*, Vol. (16)4.

Fleischanderl, G., Friedrich, G., Haselböck, A., Schreiner, H., and Stumptner, M.: 1998, Configuring Large Systems Using Generative Constraint Satisfaction, *IEEE Intelligent Systems*. Vol. 13(4).

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

Hendler, J.: Agents and the Semantic Web, *IEEE Intelligent Systems*, Vol. 16(2), pp. 46-53.

Infonetics Research, 1997. Virtual Private Networks White Paper, http://www.infonetics.com.

ILOG JConfigurator User Manual, 2001, ILOG S.A. France, 2001, http://www.ilog.com.

Junker, U.: 2001, Preference-programming for Configuration, *Proc. IJCAI'01 - Configuration Workshop*, Seattle.

Mailharro, D.: 1998, A Classification and Constraint-based Framework for Configuration, *AI EDAM*, Vol. 12(98), Cambridge University Press.

McIlraith, S., Son, T.C., and Zeng, H., 2001, Semantic Web Services, *IEEE Intelligent Systems*, Vol. 16(2), pp. 46-53.

Mittal, S. and Frayman, F.: 1989, Towards a generic model of configuration tasks, *Proc: IJCAI'89*, pp. 1395-1401.

Mowbray, T.J., and Ruh, W.A.: 1998, *Inside CORBA*, Addison-Wesley, Reading, Mass.

Peltonen, H., Männistö, T., Soininen, T., Tiihonen, J., Martio, A., and Sulonen, R.: 1998, Concepts for Modeling Configurable Products. In *Proceedings of European Conference Product Data Technology Days 1998*, Sandhurst, UK, pp. 189-196.

RosettaNet WebPages: 2001, http://www.rosettanet.org.

Rumbaugh, J., Jacobson, I., Booch, G.: 1998, The Unified Modeling Language Reference Manual, *Addison-Wesley*.

Silaghi, M., Sam-Haroud, D., Faltings, B.: 2000, Asynchronous Search with Aggregations. *Proc. AAAI/IAAI 2000*, Austin, TX, pp. 917-922.

Yokoo, M.: 2001, *Distributed constraint satisfaction - foundations of cooperation in multi-agent systems*. Springer Verlag, Berlin, Germany.

Yu, B., and Skovgaard, H.J.: 1998, A configuration tool to increase product competitiveness, *IEEE Intelligent Systems*, Vol. 13(4), pp. 34–41.