Co-Configuration of Products and On-Line Service Manuals

Carsten Sinz and **Wolfgang Küchlin**
Steinbeis Technology Transfer Center OIT
WSI for Computer Science, University of Tübingen
Sand 13, 72076 Tübingen, Germany
http://www-sr.informatik.uni-tuebingen.de
{sinz, kuechlin}@informatik.uni-tuebingen.de

Abstract

As products are growing more complex, so is their documentation. With an increasing number of product options, the diversity in service and maintenance procedures grows accordingly. Different variants of a model line consist of different components and thus require different service procedures. SIEMENS Medical Solutions has thus decided, instead of having one common on-line service handbook for all its Magnetic Resonance Tomographs, to fragment the on-line documentation into small (help) packages, out of which a suitable subset is selected for each individual product instance. Selection of help packages is controlled by XML terms encoding Boolean choice conditions. To assure that the set of available help packages is sufficient for all valid product instances, we developed a tool called *HelpChecker* that uses SAT- and BDD-based methods to guarantee completeness of the on-line documentation and to support authors in finding residual gaps.

1 Introduction

Complex products require complex handbooks. Unfortunately, product manuals traditionally encompass not only those particular features the customer ordered, but all possible components of the whole model line. Over the last years, many products (e.g. cars, computers, telecommunication equipment, medical devices) have seen a tremendous increase in the number of available product options, the handbooks grew in size accordingly, thus making matters even worse.

Ideally, each product instance would be equipped with its own product manual, covering only the functionality that is in fact relevant for that product instance. However, this approach is still not very common, perhaps due to the fact that it would require a configuration of the manuals themselves in conjunction with the configuration of the product.

SIEMENS Medical Solutions recently introduced a new XML-based product configuration system for their Magnetic Resonance (MR) Tomographs. In the course of this introduction, they decided to equip their products with individually configured on-line service manuals. Now, the material of all handbooks is split up into smaller help packages, each covering a clearly delimited topic. Each service handbook is then automatically generated as a suitable selection of those packages. Selection of packages is controlled by Boolean logic formulae encoded as XML terms, and during configuration of a product instance its manual is configured simultaneously.

The authors of the help packages can decide autonomously about how to break down the whole help documentation into smaller packages. So it is their own decision whether to write a whole bunch of smaller packages, one for each system configuration, or to integrate similar packages into one. Writing of service manuals does not follow the configuration structure of each individual system but is done for similar service situations across all product lines. However, this approach makes it hard to determine whether all needed help packages already have been written or whether there are cases that still need to be covered. To guarantee completeness of the manuals for all possible, valid product instances a final cross-check is needed. We therefore developed a tool called *HelpChecker*, which informs the authors of the documentation department which parts of the manual they still have to write.

In the following, we describe the method used by SIEMENS for documenting their MR products, how service manuals are associated with product instances, and how we check the set of all help packages for completeness with our add-on tool *HelpChecker*.

2 Configuring SIEMENS MR Tomographs

Many different formalisms have been proposed to model the structure of complex products [Mittal and Frayman, 1989; Sabin and Weigel, 1998; Soininen *et al.*, 1998; McGuiness and Wright, 1998; Küchlin and Sinz, 2000]. The method used by SIEMENS for the configuration of their MR systems was developed in collaboration with the first author of this paper and resembles the approach presented by Soininen *et al.* [Soininen *et al.*, 1998]. Structural information is explicitly represented as a tree. This tree serves two purposes: first, it reflects the hierarchical assembly of the device, i.e. it shows the constituent components of larger (sub-)assemblies; and, second, it collects all available, functionally equivalent configuration options for a certain functionality. These two distinct purposes are reflected by two different kinds of nodes in the tree, as can be seen from the example in Fig. 1: *Type Nodes* represent configurable entities and are employed to

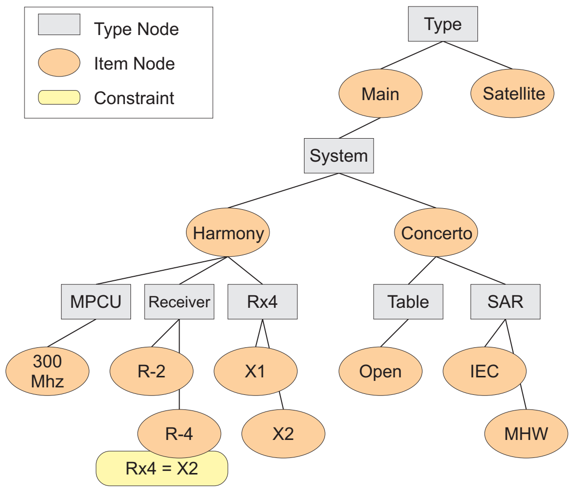


Figure 1: Product Structure of Magnetic Resonance Tomographs (Simplified Example).

reflect the hierarchical structure of the product; *Item Nodes* represent individual configuration options. The two types of nodes appear strictly alternatingly on each path of the tree. All of a Type Node's children thus are Item Nodes and vice versa. Each Item Node is one configuration possibility for the parent node.

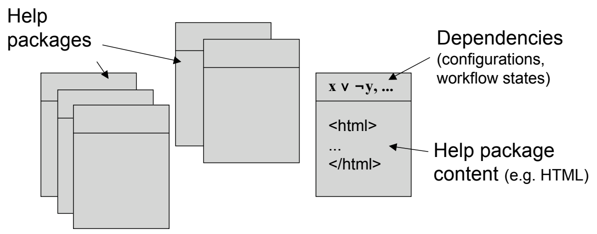
From the example tree shown in Fig. 1 we may, e.g., conclude that there are two alternatives for choosing a *System*: *Harmony* and *Concerto*. A *Harmony* system possesses three configurable (direct) subcomponents, of type *MPCU*, *Receiver*, and *Rx4*, respectively. The receiver, in turn, may be selected from the two alternatives *R-2* and *R-4*. Choosing the latter option puts an additional restriction on the configurable component *Rx4*: this has to be selected in its form *X2*. Each type node possesses additional attributes *MinOccurs* and *MaxOccurs* to bound the number of subitems of the respective type to admissible values. Assuming that for each type exactly one item has to be selected (i.e. $MinOccurs = MaxOccurs = 1$ for all type nodes), the configuration tree shown in Fig. 1 permits, e.g., the following valid configuration (set of assignments):

$$\begin{array}{cc} \text{Type} = \text{Main} & \text{System} = \text{Harmony} \\ \text{MPCU} = \text{300MHz} & \text{Receiver} = \text{R-4} \\ \text{Rx4} = \text{X2} & \end{array}$$

3 Configuring Service Manuals

Within the SIEMENS system, the tree describing all product configurations is represented as an XML term. The term corresponding to the tree of Fig. 1 is shown in Fig. 4. All XML terms are checked for well-formedness using XML Schemas [XMLSchema, 2001]. Part of the XML Schema describing valid XML terms for product configurations is shown in Fig. 5.

The on-line help pages that are presented to the user of an MR system may depend on the configuration of the system. For example, help pages should only be offered for those components that are in fact present in the system configuration. Moreover, for certain service procedures (e.g., tune up, quality assurance), the pages depend not only on the system configuration at hand, but also on the (workflow) steps that the service personnel already has executed. Thus, the help system depends both on the configuration and the state of the workflow.



Help packages
Dependencies
(configurations, workflow states)
x ∨ ¬y, ...
<html>
...
</html>
Help package content (e.g. HTML)

Figure 2: Illustration of Help Packages: For each system configuration and workflow state a suitable help package has to be selected (controlled by dependencies).

To avoid writing the complete on-line help from scratch for each possible system configuration and all possible workflow states, the whole help system is broken down into small *Help Packages* (see Fig. 2). A help package contains documents (text, pictures, demonstration videos) on a specialized topic. The authors of the help packages decide autonomously about how they break down the whole help document into smaller packages. So it is their own decision whether to write a whole bunch of smaller packages, one for each system configuration, or to integrate similar packages into one.

```
<Package ID="HLP_HP-1-181203-01-001" Name="HP-1-...">
   <Content> ... </Content>
   <Dependencies>
      <Dependency>
         <RefType IDREF="INT_Workflow">
            <RefItem IDREF="INI_Workflow_TUNEUP"/>
         </RefType>
         <RefType IDREF="INT_System">
            <RefItem IDREF="INI_System_003"/>
         </RefType>
      </Dependency>
   </Dependencies>
</Package>

<Context>
   <RefType IDREF="INT_System">
      <RefItem IDREF="INI_System_003"/>
   </RefType>
   <RefType IDREF="INT_Workflow">
      <RefItem IDREF="INI_Workflow_TUNEUP"/>
   </RefType>
   <RefType IDREF="INT_WorkflowMode">
      <RefItem IDREF="INI_WorkflowMode_General"/>
   </RefType>
   <RefType IDREF="INT_WorkflowSfp">
      <RefItem IDREF="INI_WorkflowSfp_SfpTuncalOpen"/>
   </RefType>
</Context>
```

Figure 3: Example of a Help Package (with Dependencies) and a Help Context.

Now, in order to specify the assignment of help packages to system configurations, a list of *dependencies* is attached

```
<Config auto-ns1:noNamespaceSchemaLocation="Config.xsd"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <Structure>
      <Type IDREF="INT_ConsoleType" MinOccurs="1" MaxOccurs="1">
         <Item IDREF="INI_ConsoleType_Sat"/>
         <Item IDREF="INI_ConsoleType_Main">
            <SubType IDREF="INT_System" MinOccurs="1" MaxOccurs="1">
               <!-- Harmony -->
               <Item IDREF="INI_System024">
                  <SubType IDREF="INT_Comp_MPCU" Default="INI_Comp_MPCU300" ReadOnly="true" MinOccurs="1" MaxOccurs="1">
                     <Item IDREF="INI_Comp_MPCU300"/>
                  </SubType>
                  <SubType IDREF="INT_Comp_RXNumOf" Default="INI_Comp_RXNumOf1" MinOccurs="1" MaxOccurs="1">
                     <Item IDREF="INI_Comp_RXNumOf1"/>
                     <Item IDREF="INI_Comp_RXNumOf2"/>
                  </SubType>
                  <SubType IDREF="INT_Comp_ReceiverNumOf" MinOccurs="1" MaxOccurs="1">
                     <Item IDREF="INI_Comp_ReceiverNumOf2"/>
                     <Item IDREF="INI_Comp_ReceiverNumOf4">
                        <Conditions>
                           <Condition Type="INT_Comp_RXNumOf" Op="eq" Value="INI_Comp_RXNumOf2"/>
                        </Conditions>
                     </Item>
                  </SubType>
               </Item>
               <!-- Concerto -->  ...
            </SubType>
         </Item>
      </Type>
   </Structure>
</Config>
```

Figure 4: Product Structure Tree of Fig.1 in XML Representation (Excerpts).

to each help package, in which the author lists the system configurations and workflow states for which his package is suitable (see Fig. 3 for an example): all of a dependency's `RefType`/`RefItem` assignments must match in order to activate the package and to include it into the set of on-line help pages for that system. Multiple matching situations may be specified by associating further `Dependency`-elements with the package.

The situations for which help packages must be available are specified by the engineering department using so-called *Help Contexts*. A help context determines system parameters and workflow steps for which a help package must be present. Examples for both a help package and a context (in XML representation) can be found in Fig. 3. The help package of this example fits any state of workflow *tune up* and all configurations of *System_003*. The example's context specifies that for step *TuncalOpen* in the *tune up* procedure for *System_003* a help package is required, in case the workflow mode is set to *General*.

Currently, almost a thousand help contexts have been defined for eleven MR systems (model lines), each with millions of different configuration possibilities. So, in spite of in-depth product knowledge, it is a difficult and time consuming task for the authors of help packages to find gaps (missing packages) or overlaps (ambiguities in package assignment) in the help system. To assist the authors, we therefore developed the *HelpChecker* tool, which is able to perform cross-checks between the set of valid system configurations, the situations for which help may be requested (determined by the contexts) and the situations for which help packages are available (determined by the packages' dependencies).

4 Checking Completeness and Consistency

Our implementation *HelpChecker* is a C++ program that builds upon Apache's Xerces XML parser to read the SIEMENS MR product structure and help package dependencies. From these data, it generates two propositional logic formulae:[1]

(A) **Completeness of the help system**. If this formula is valid, then for all valid MR configurations and all situations in which help may be requested, at least one matching help package is specified (via its dependencies).

(B) **No overlaps between help packages**. If this formula is satisfiable, there is a valid system configuration and a situation for which help may be requested (by a context) with more than one matching help package: an overlap exists (i.e. a fault in the help package assignment).

After having generated these formulae, they are checked for validity resp. satisfiability using BDD-based methods. In case of an error condition, a formula is generated describing the set of situations in which this error occurs. This formula, call it F, is simplified by existential abstraction over irrelevant variables using standard BDD techniques (i.e. by replacing F by $\exists x F$ or, equivalently, by $F|_{x=0} \vee F|_{x=1}$ for an irrelevant propositional variable x).

HelpChecker is embedded into a larger interactive authoring system for the writers of the help packages at SIEMENS. The authoring system offers a graphical user interface, by which the help packages' content and dependencies can be entered and maintained. The overall structure of the handbooks as well as the hyperlinks between different pages of

[1]Details on how these formulae are generated can be found in another publication [Sinz and Küchlin, 2004].

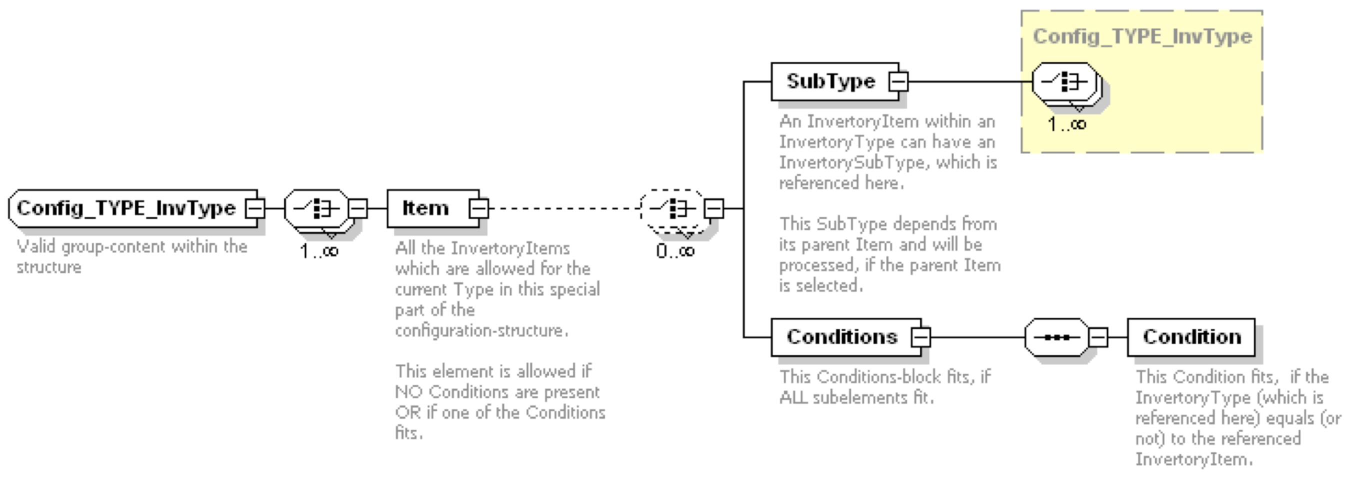

Config_TYPE_InvType
Valid group-content within the structure
1..∞
Item
All the InvertoryItems which are allowed for the current Type in this special part of the configuration-structure.
This element is allowed if NO Conditions are present OR if one of the Conditions fits.
0..∞
SubType
An InvertoryItem within an InvertoryType can have an InvertorySubType, which is referenced here.
This SubType depends from its parent Item and will be processed, if the parent Item is selected.
Config_TYPE_InvType
1..∞
Conditions
This Conditions-block fits, if ALL subelements fit.
Condition
This Condition fits, if the InvertoryType (which is referenced here) equals (or not) to the referenced InvertoryItem.

Figure 5: Graphical representation of the part of the XML-Schema that deals with the product structure.

the documentation are pre-specified by the technical department and cannot be altered by the help package authors. This also means that there is no intra-page hypertext navigation that is under control of the help checker. Versioning of products and handbooks is supported by the underlying XML data base schemas. In the authoring tool editors have to select a special version of the data they are working on, however. This also means that help packages for a follow-up version of the documentation may only be copied from an older version, but there is no way to specify (by constraints) that they are also valid for the new edition.

Production use of the system is just starting, thus we do not have user feedback yet, and can only report on manually created test cases (that were generated by SIEMENS documentation experts, though). These tests contained eleven basic MR systems, 964 help contexts and twelve (dummy) help packages. The BDDs reflecting test formulae (A) and (B) contained up to 9715 nodes and 458 variables (with intermediate BDD sizes during formula generation of over 100,000 nodes).[2] A complete check of our experimental XML help—including computation of error situations—took 6.96 seconds. The input data we used already contain the complete configuration structure for all of SIEMENS' current MR model lines. However, the number of help packages is much lower than what we expect during production use.

5 Conclusion

In this paper we have shown that individually configured service handbooks are feasible. By using a propositional logic encoding for both system configurations and the assignment of handbook chapters (help packages) to product instances, we were able to employ advanced Boolean logic techniques (like BDDs). Those turned out to have enough potential to handle the problems brought up by our completeness and consistency checks. Although we have demonstrated the feasibility of our method only for the MR systems of SIEMENS Medical Solutions, we suppose that the presented techniques are also usable for other complex products. More generally, we expect that a wide range of cross-checks between XML documents can be computed efficiently using propositional logic techniques.

[2]We have also conducted experiments using a SAT-Solver instead of the BDD package, and obtained very promising results.

References

[Küchlin and Sinz, 2000] W. Küchlin and C. Sinz. Proving consistency assertions for automotive product data management. *J. Automated Reasoning*, 24(1–2):145–163, February 2000.

[McGuiness and Wright, 1998] D.L. McGuiness and J.R. Wright. Conceptual modelling for configuration: A description logic-based approach. *AI EDAM*, 12(4):333–344, 1998.

[Mittal and Frayman, 1989] S. Mittal and F. Frayman. Towards a generic model of configuration tasks. In *Proc. of the 11th Intl. Joint Conf. on Artificial Intelligence*, pages 1395–1401, Detroit, MI, August 1989.

[Sabin and Weigel, 1998] D. Sabin and R. Weigel. Product configuration frameworks – a survey. *IEEE Intelligent Systems*, 13(4):42–49, July/August 1998.

[Sinz and Küchlin, 2004] Carsten Sinz and Wolfgang Küchlin. Verifying the on-line help system of SIEMENS magnetic resonance tomographs. In *Proc. of the 6th Intl. Conf. on Formal Engineering Methods (ICFEM'2004)*, pages 391–402, Seattle, WA, November 2004. Springer.

[Soininen *et al.*, 1998] T. Soininen, J. Tiihonen, T. Männistö, and R. Sulonen. Towards a general ontology of configuration. *AI EDAM*, 12(4):357–372, 1998.

[XMLSchema, 2001] *XML Schema Parts 0–2: Primer, Structures, Datatypes. W3C Recommendation*, May 2001.