International Journal of Mechanical, Industrial and Aerospace Sciences
ISSN: 2517-9950
Vol:1, No:1, 2007

Multimodal Reasoning in a Knowledge Engineering Framework for Product Support

Rossitza M. Setchi, and Nikolaos Lagos

***Abstract*—**Problem solving has traditionally been one of the principal research areas for artificial intelligence. Yet, although artificial intelligence reasoning techniques have been employed in several product support systems, the benefit of integrating product support, knowledge engineering, and problem solving, is still unclear. This paper studies the synergy of these areas and proposes a knowledge engineering framework that integrates product support systems and artificial intelligence techniques. The framework includes four spaces; the data, problem, hypothesis, and solution ones. The data space incorporates the knowledge needed for structured reasoning to take place, the problem space contains representations of problems, and the hypothesis space utilizes a multimodal reasoning approach to produce appropriate solutions in the form of virtual documents. The solution space is used as the gateway between the system and the user. The proposed framework enables the development of product support systems in terms of smaller, more manageable steps while the combination of different reasoning techniques provides a way to overcome the lack of documentation resources.

***Keywords*—**Knowledge engineering framework, product support, case-based reasoning, model-based reasoning, multimodal reasoning.

I. Introduction

PRODUCT support is often described through the various forms of assistance that the companies offer to their customers. Traditionally, it is associated with the provision of supplies, tools, equipment and facilities, as well as information. It may include installation, user training, technical documentation, product manuals, help lines, servicing, spare parts, maintenance management, and product upgrades [1], [2].

Nowadays, efficient support involves answering to users' queries by providing accurate and user-tailored information. Each user query can be represented as a problem, using approaches from problem solving research. The diversity of the problems posed to the system should be an essential consideration throughout its design.

The aim of this paper is to demonstrate that product support problems can be automatically solved by deploying artificial intelligence representation and reasoning techniques in a knowledge engineering framework. The rest of the paper is organised as follows. Section 2 briefly reviews product support and its relation to knowledge engineering. The knowledge engineering framework, which comprises of four different spaces, is introduced in section 3. Section 4 includes a case study while the final section contains conclusions and directions for further work.

Manuscript received October 31, 2006. This work was supported in part by the ISAR and TRENDS European projects.

R. M. Setchi is with Cardiff School of Engineering, Cardiff University, Cardiff CF24 3AA (phone: +44-(0)-292087-5720; fax: +44-(0)-292087-4716; e-mail: setchi@cf.ac.uk).

Nikolaos Lagos is with Cardiff School of Engineering, Cardiff University, Cardiff CF24 3AA (e-mail: LagosN@cf.ac.uk).

II. Background

Pham et al. [1] define product support as everything necessary to allow the continued use of a product. It takes various forms, ranging from conventional paper-based technical manuals to more advanced interactive electronic technical manuals (IETMs) [3], intelligent product manuals (IPMs) [1] and electronic performance support systems (EPSSs) [4].

Studies show that the most common AI technique used in product support systems is rule-based reasoning, which is primarily employed in troubleshooting. An example is the work of Paul et al. [5], in which a diagnostic system supports the operation of a radar warning receiver.

In addition, a number of researchers have used case-based reasoning (CBR) for diagnosis and help-desk applications. Foo et al. [6] utilise CBR in combination with neural networks for producing a help-desk-support environment, while Auriol et al. [7] use a CBR system in the troubleshooting of a welding robot.

Model-based reasoning has received less attention than these two techniques. An example is the research of Brusilovsky and Cooper [8], who employ models for adapting the interface of a performance support system and creating an 'expert-like' problem solving engine.

Latest attempts focus on the integration of different reasoning techniques. For instance, Pham and Setchi [9] develop adaptive product manuals by combining CBR for interpreting user's requests and rules for adapting the generated documents.

Although these studies address the use of reasoning techniques in product support they are based on ad-hoc designs. As a result, a major limitation of the previous work is the lack of design and knowledge reusability. This could become an obstacle in the nearest future when a new generation of much more complex and highly customized products emerges. The authors of this research share the vision that this challenge could be successfully addressed if

the synergy between problem solving, knowledge engineering, and product support is further studied. The research reported in this paper extends earlier approaches by proposing a framework that delineates the inference components of a product support system and deploys a multimodal reasoning approach to enable the segmentation of the development process into smaller more manageable steps.

III. A Knowledge Engineering Framework for Product Support Systems

The framework (Fig. 1) comprises four different spaces: the data, problem, hypothesis, and solution ones. The architecture of the framework is sequential, meaning that each space is involved in the process, when its preceding space has completed its operations. The sequential structure of the framework is enhanced with several feedback paths, which enable other advanced operations to take place, such as knowledge creation.

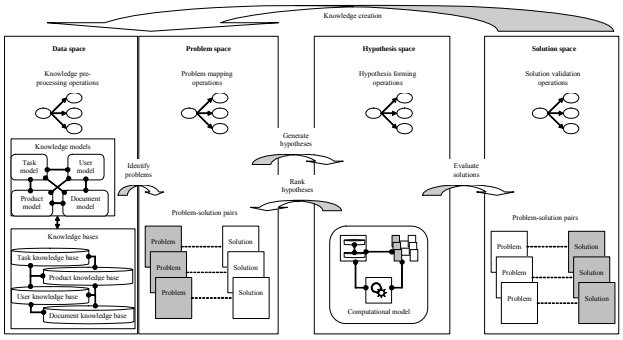

Fig. 1 Knowledge-engineering framework for product support

A. Data Space

The data space contains knowledge about the domain (task and product models), the context (e.g. activity model, user model) and the documentation elements. As a result, the development of such a system is highly interdisciplinary. In order to advance interoperability between these different areas and product support, an ontology that formalizes the aforementioned knowledge has been developed (Fig. 2).

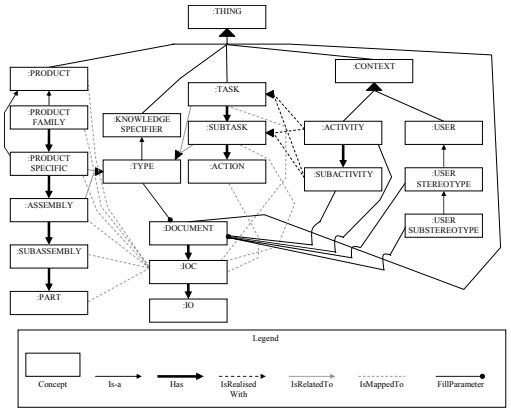

Fig. 2 Part of the ontology for product support systems

The ontology can be described according to the product, task, context, and document models it includes.

1) Document model

The main structural component of the document model is the Information Object (IO). IO is defined as "a data structure that represents an identifiable and meaningful instance of information in a specific presentation form" [10]. IO can therefore be a picture that illustrates a part of a product or a textual description.

The notion of Information Object Cluster (IOC) has been utilized [11] as a means of organizing IOs. IOC is defined as a 2-tuple IOC:=({IO}, S_{IOC}) where {IO} is a set of IOs sharing a common property that are arranged in a structure S_{IOC}. A structure defines the way in which they are presented within the same page, as well as the relevant links. S_{IOC} conforms to presentation rules (e.g. a textual description should always appear before the corresponding image).

The Virtual Document (D) is generated by the aggregation of IOCs and is defined as a 2-tuple D:=({IOC}, S_D) where {IOC} is a set of IOCs sharing a common property that logically structured (S_D) in order to compose a document (D).

2) Product model

The product model represents the structure of the product. All its concepts are mapped to IOCs as explained in the rest of the section. Concepts: "PRODUCT SPECIFIC" and ":ASSEMBLY" are linked to the concept ":TYPE". This is a specialization of: "KNOWLEDGE SPECIFIER", which abstracts all concepts that represent domain significant properties. For example, the type of an assembly, i.e. whether it is considered as complex or not, affects the generation of the document (this will not be further discussed here as it is not within the focus of this paper).

3) Task model

The task model contains the tasks, subtasks, and actions that are supported, where action is the most elementary step of a task. All are mapped respectively to IOCs and are related to ":TYPE". Furthermore, the task model is configured according to its relation to the activity model.

4) Context model

The context model includes the activity and user models (it can include their parameters such as location). Both models are related to the ":DOCUMENT" concept with the relation "FillParameter", which denotes that the characteristics of the user and the activity are passed as adaptation parameters to the document.

The ontology is not in the focus of this paper. Please refer to [11] for more information.

B. Problem Space

The problem space incorporates knowledge about the product support problems (PSPs) that have occurred in previous problem solving iterations. In order to automatically solve such problems, PSPs have first to be formally defined and appropriately represented in a machine-processable format.

1) Definition of PSPs

The knowledge that a product support system should deliver to the user is tightly linked to the problem that has to be solved. There are two basic qualities that characterise a PSP, namely its content and context. The content has to be relevant to the product that is supported and/or the task that the user wants to perform, while the context is determined according to the user characteristics and the system's usage. The PSP therefore should contain what is needed (elements of knowledge that are missing), why it is needed and under what circumstances (context). The given definition of a PSP contains all the identified elements and is represented as follows.

Definition 1. P*roduct Support Problem* (PSP) is a 4-tuple PSP := (MOD, HYP, CON, OBS) where:

- *MOD* is a finite set that represents the product and task models in relation to the IOCs and IOs that form the documents.
- *HYP* is a finite set of combinations of elements of MOD representing possible documentation hypotheses.
- *CON* is the context that characterises the problem and contains the user model (UM) in combination with the usage purpose.
- *OBS* represents the observations acquired by the current query and are mapped to elements of MOD and CON.

Definition 1 identifies PSPs as a specialisation of diagnostic problems, since a product support system recognises and solves PSPs in terms of the IOs and IOCs involved. The problem solving process therefore includes identifying that there is a fault (e.g. product support virtual document asked does not exist) recognising the type of fault (e.g. difference in configuration or missing IO, IOC), and choosing a strategy to be followed (e.g. provide the missing documentation element). In order to achieve that PSPs have to be appropriately represented.

2) Case-based representation of PSPs

PSPs are literally considered as problem-solution pairs, as they are directly linked to solutions that exist in the solution space. A natural way to represent problem-solution pairs is that of cases.

Cases are structured in the form of attribute-value pairs. This form of representation has the advantage of simplicity, preciseness, and controllability (important characteristics for a system used by different groups of users e.g. novice in information or web technologies). For example, the attribute "No_Disks", which is used to describe automotive clutches, can be paired with the value "2". The assignment indicates that the problem refers to double-disk clutches.

The content of a case-based problem contains the goals to be achieved, the situation description, and the constraints to be satisfied. The **goals** are separated in three groups. According to the definition of a product support problem the most abstract goal is to execute diagnosis (explicitly related with the use of a product support system), which does not have to be included in the problem description. For example, identifying that specific parts of required product support virtual documentation are missing and utilising the means to fix this problem belongs in this category.

At the next group the purpose of the user is delineated, into three classes, which are information retrieval, diagnosis, and explanation (or expert advice). These can be implicitly identified, according to the usage of the system. They indicate the type of information the user requires for the supported products and tasks. For example, the query "Loud bang or chattering is heard as vehicle vibrates" belongs to the diagnosis class since the goal is to diagnose the behaviour of the supported product, while "give more information on clutches" relates to information retrieval.

The last group differentiates between educational (i.e. knowledge enrichment) and performance (i.e. increased efficiency) objectives. This group of goals influences the way the documentation content is delivered to the user.

The **situation** components give descriptive information about the targeted characteristics that the solution should reflect. For example the dimension[1] "Moment_of_Inertia" with value "55.814", depicts information about the performance of a clutch. However, although desirable, respecting the restrictions set by such descriptors is not deemed necessary for delivering a solution. Such features are therefore set to contain either highly dynamic values (e.g. the value of the moment of inertia theoretically can range from 0, in case the product does not have any mass or radius, to several hundreds depending on the supported clutches) or static values that have not been included in the query (e.g. if the query is "more information on clutches" then whether the clutch contains a synchroniser or not should not disallow the presentation of a product support document for clutches).

The **constraints** are conditions set on goals that have to be met in any acceptable solution. For example one of the goals is to diagnose the documentation constituents needed and deliver them (through a product support virtual document) to the user. If the query asks for more information on transaxles, then the presented product support virtual document should include such information, otherwise the system fails. Class-level and contextual features that are engaged in the query form constraints. Contextual features describe the user's category and goal.

3) Integrating case-based PSPs and ontologies

One of the major limitations of traditional attribute-value pair representations of cases is the fact that there is no relation between the different pairs.

In this study the aforementioned drawback is leveraged by means of modifying the weight of each feature. However, since the CBR knowledge is most of the time stored in text format having no identified links between a large number of attribute-value pairs can influence the performance of a system in case retrieval and adaptation.

The proposed solution is semantic-based disambiguation of the features by assigning them to components of product

[1] The terms *descriptor* and *feature* apply to the attribute-value pair, while *dimension* is the attribute part of the pair only.

support knowledge bases and ontologies. The mapping is achieved in two levels, the concept and instance ones.

At the **concept level** the dimension of a feature is mapped to either a concept or a slot. For example, the attribute "Assembly" is assigned to the assembly concept in the product knowledge base, while the dimension "Moment_of_Inertia" represents the same information as the corresponding slot. The aforementioned allocation has the following repercussions.

- The descriptors of the cases are associated with each other according to the relations of the corresponding knowledge models elements. For example, the dimension "No_of_Disks" is related to the concept "Clutch" with the relation "is_slot", which means that the value of the "Assembly" feature has to be "Clutch", when the descriptor "No_of_Disks" belongs to the case, setting restrictions on the validity of cases. In natural language this can be expressed as "The clutch assembly has number of disks (value)".
- The features can be classified according to the range of values they can have or the frequency with which their values are expected to change. Dimensions that denote slots are expected to demonstrate dynamism (e.g. "No_of_Disks" can change frequently within the range specified by the knowledge base), while concept-based descriptors tend to be more static and predictable (e.g. "Assembly" can have only pre-specified values that correspond to concepts in the knowledge base). The former group of features is called *variable-level* while the latter is named *class-level*. The values of class-level features are always concepts. The difference in the possible values that the sets of descriptors can take indicate that separate strategies that need to be used for accommodating modifications in variable-level and class-level features.
- The sets of attribute-value pairs can be linked to different documentation components. As already explained, concepts are described by Information Object Clusters (IOCs), while Information Objects (IOs) are mapped to slots of the knowledge base. Naturally, since variable level features are related to slots of the knowledge base they are also described by IOs and class-level descriptors by IOCs.
- Class-level features represent a more complex documentation module than variable-level ones. This means that the modification of class-level descriptors requires a lot of computational resources and knowledge-intensive techniques, in order to produce a support document. On the other hand changing variable-level descriptors is less important since they are mapped to the smallest documentation constituent (i.e. IO). This distinction is signified by the weights assigned to each group. Consequently, class-level features should have a bigger weight factor than variable-level ones, unless otherwise explicitly defined by the user.

At the **instance level** the cases contained in the case base represent combination of instances included in the knowledge base. For example, if two features of a case are "Assembly" and "No_of_Disks" with respective values "Clutch" and "2", then instances of the concept clutch with 2 disks should exist in the knowledge base. Variable-level descriptors are directly related with the validity of cases, since they are the ones used to instantiate concepts.

C. Hypothesis Space

The formal definition and semantic representation of PSPs in terms of cases, enables the automatic creation of solutions as described in the rest of this section. The process followed includes retrieving relevant cases, adapting them and/or generating new ones.

1) Case retrieval

The functional roles of the case components are well defined and simple comparison of values that correspond to the same dimension is feasible. A weighted ranking method therefore can be used to input the degree of importance for dimensions. The weighted Euclidean method has proved to perform better than other techniques for certain applications [12]. In this technique all features of the cases are represented as vectors. The following formula describes the weighted Euclidean distance.

$$D = \sqrt{\sum_{i=1,\ldots,n} w_i (q_i - c_i)^2} \quad (1)$$

In (1) D stands for distance, w_i depicts the weight of the descriptor i, q_i the target query, and c_i the compared case.

2) Case-based adaptation

Variable-level descriptors can have different values. For example, "radius" is a dynamic characteristic, which requires a solution modification when changed from 2.2 cm to 3.4 cm that reflects the current situation. Most of the times such a variation does not affect the structure of the presented PSVD. **Parameter adjustment** is utilised for enabling adaptation based on modified variable-level descriptors.

Class-level descriptors may be also altered. Case-based adaptation is employed in such a case if the Information Object Cluster that substitutes an existing solution's IOC, has the same functional role and is pre-composed or can be composed at run-time (IOs are available and structure can be determined). For example an IOC that describes the flywheel of a clutch has the same role as an IOC depicting a countershaft (both are clutch subassemblies). In the case that both of them are available and the countershaft IOC is required to be included in the solution instead of the flywheel IOC, then case-based adaptation can be applied. In the majority of such cases not only the content of the PSVD needs to change but also its structure. **Reinstantiation** is used in such cases by selecting an old solution and employing role bindings for creating an adapted solution. For example the two cases illustrated in Fig. 3 involve requesting information about a flywheel and a countershaft (according to the bindings between the ontology and the case base "flywheel" and "countershaft" are both specialisations of the subassembly concept and therefore semantically equivalent for the case reasoner) and can be used to abstract the problem of asking information about subassemblies, as shown in the right part of

the figure.

3) Model-based generation

Model-based reasoning is utilised in the case when a class-level descriptor that is included in the PSP, does not correspond to an existing IOC. The reasoning process has two main stages in such situations, diagnosis and configuration.

The goal of **diagnosis** is to isolate the fault to a single component or to a least replaceable unit (LRU). Since the repair action is to construct a new IOC, the LRU denotes an IOC, although the IOC is in turn composed of smaller constituents (i.e. IOs). Identifying the omitted IOC is a matter of exploring the model of a case description and connecting it to the ontology-based knowledge models.

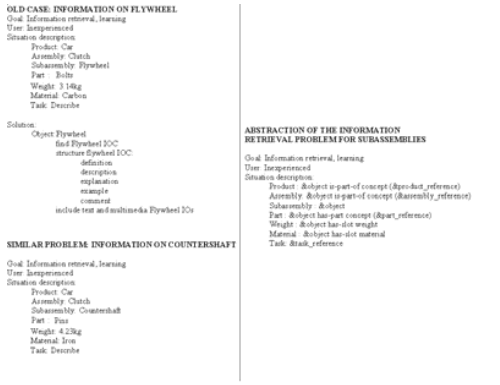

Fig. 3 Abstracting groups of cases to represent information retrieval problems based on ontology-related bindings

Once the required LRU is mapped to an ontology component, the **configuration** process starts by automatically creating an IOC based on the relations defined in the knowledge base. More specifically, the generalisation relation ("is-a") is utilised in providing information that covers the queried domains, while aggregation and reference relations are employed to compose required IOCs.

For example, if an IOC that corresponds to a transaxle is needed but no such description is available, then aggregation relations are exploited to find the assemblies and parts with which a transaxle is developed. Each assembly, part, and relation is individually used to portray the description of a transaxle. In the case that transaxle is not recognised as an internal part of the model, the concept related to it via a generalisation relation (e.g. assembly) is used to retrieve information based on the general qualities that characterise the domain (e.g. transaxle is-a assembly and therefore the definition of assemblies is true for transaxles as well).

Candidate solutions are ranked according to a specialisation of the parsimony rule (i.e. if a composite solution is a data subset of another one, then the smallest set is selected). For example, a transaxle has a number of subassemblies, which in turn have a number of parts. At least two hypotheses are formed in such a case. The first one has data about the subassemblies only, while the second one about both subassemblies and parts. If the IOCs that describe the subassemblies have been manually pre-composed then the former hypothesis is selected otherwise the latter one is chosen.

D. Solution Space

The solution space contains information about the product support virtual documents that have been derived throughout previous problem solving iterations and includes unique identifiers (UIs) and status identifiers (SIs).

Each UI comprises the concepts and slots that were involved in the problem specification and solution generation as shown below. All instantiations denote the existence of a

Goal-User-Product-ProductInstantiation-Assembly-AssemblyInstantiation-Subassembly-SubassemblyInstantiation-Part-PartInstantiation-Task-Subtask-Action

number of variable-level features (or ontology slots). The parts of the UI that do not have any value are filled with "null" or "0".

Each SI can take the values "validated" or "not validated", which indicate whether each solution has been manually validated or not. The ones that have not been validated should be removed after a period of time.

IV. Case Study

The selection of simple retrieval, case-based adaptation or model-based generation depends on the cases retrieved as a response to the queries of the users. Fig. 5(A) illustrates an example of a pre-composed document that describes the clutch assembly. The first picture included in Fig. 5(A) corresponds to single-disk clutches while the second picture portrays double-disk clutches. Fig. 5(B) shows a scenario where the user asks for a description of a single-disk clutch. That is expressed by the highlighted row in Fig. 5(B), which is one of the retrieved cases that match the user's query. The resultant virtual document includes only the documentation components related to single-disk clutches such as the related pictures and facts (denoted by the arrows), produced with parameter adjustment heuristics.

The next scenario involves a user who requests a description for the body of a car. The query can be satisfied by replacing the *Clutch* concept from the previous case with the *Body* concept (represented by the values of the "ASSEMBLY" dimension in the cases) since both of them are considered specialisations of the *Assembly* concept in the product ontology. Reinstantiation is therefore utilised and the IOC related to the *Clutch* is replaced by the IOC describing the *Body* of a car, in order to respond to the changes between the previous and current selected cases. The resultant document is shown in Fig. 5(C).

A rather similar query is examined in the following setting where the user presumably asks for the description of a transaxle. However, in this scenario there is no pre-composed IOC corresponding to the *Transaxle* concept. This means that the IOC and PSVD content has to be created according to the ontology-related models. A simple solution is to utilise the

aggregation relations, as discussed earlier, between the subassemblies and parts that compose a transaxle additionally to other information found in the slots of each concept and the information extracted from the generalisation relations. For example, in Fig. 6 the text describing the *Housing* concept is a combination of its relation to the *Subassembly* concept (i.e. "Housing **is a** Cls(**Subassembly**, FrameID(1:10133))"), and to the parts *Screw, Pin,* etc. ("**Has** Screw, Pin, … parts").

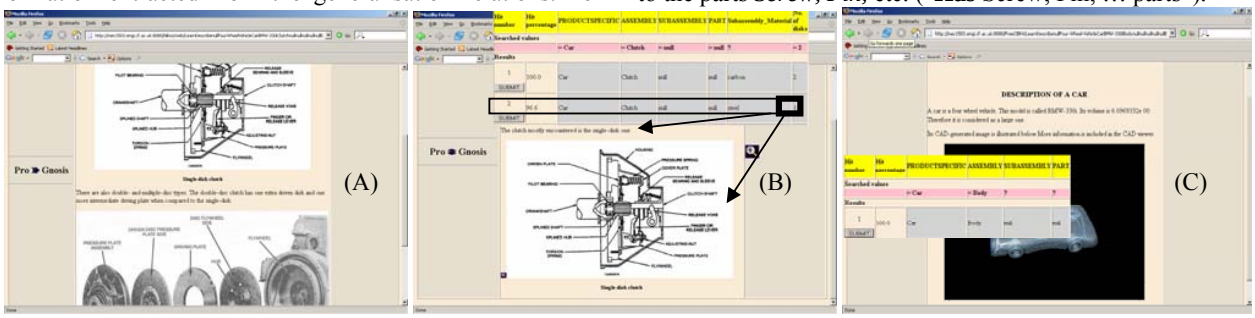

Fig. 4 Case-based adaptation

Furthermore, the definition of *Housing* is included in the *Documentation* slot and used in the generated document. As illustrated, the generated document follows the basic structure of a PSVD having title ("DESCRIPTION OF A TRANSAXLE") and introduction ("In four wheel vehicles and specifically in cars") related to the query. The body of the document is a composition of subassembly and part related IOCs. Frame IDs are included in the produced document in order to indicate the IOCs that need to be manually edited by a technical writer at a later stage (to aid in the authoring process). The assumption is that model-based generated solutions are going to become fewer as more documentation components are developed by the technical writers while case-based adaptation will become more important.

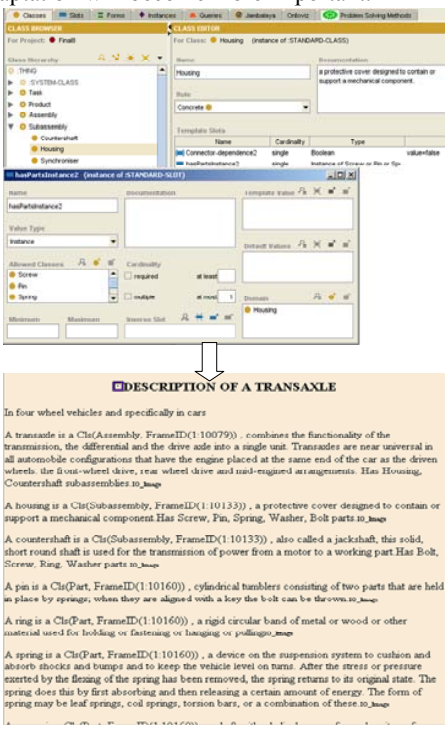

Fig. 5 Model-based generation

V. Conclusion and Future Work

The framework presented in this paper structures the task of creating a product support system in four different distinct phases. First the knowledge required to enable reasoning is represented in the data space and includes product, task, documentation, and context related information. Then possible scenarios in the form of cases in a case base stand for different types of product support problems. Case-based adaptation and model-based generation are both needed in order to respond to different user queries and deliver solutions in the form of automatically created virtual documents.

One crucial issue is the modeling of the different context instantiations and the way these can influence the document adaptation and creation. In the current work the context is included in the definition of product support problems and in the ontologies underlying the framework. The next step is to identify the correlation between virtual documentation and context in order to provide context-aware product support.

References

[1] Pham DT, Dimov SS and Setchi RM. Intelligent product manuals. Proc. IMechE. I-213 (1999) 65-76.
[2] Goffin K. Evaluating customer support during new product development-an exploratory study. Journal of Production Innov. Management. 1-15 (1998) 42-56.
[3] Jorgensen EL and Fuller JJ. A web-based architecture for interactive electronic technical manuals (IETMs). ASNE Naval Logistics Conference. (1998).
[4] Bezanson WR. Performance support: online, integrated documentation and training. 13th ACM Annual Conference on Emerging from Chaos. Savannah. USA. 1-10 (1995).
[5] Paul C, Zeiler G and Nolan M. Integrated Support System for the Self Protection system. IEEE Systems Readiness Technology Conference Proceedings, AUTOTESCON (2003) 155-160.
[6] Foo S, Hui SC and Leong PC. Web-based Intelligent Helpdesk-support Environment. Int. J. Syst. Sc. 33-6 (2002) 389-402.
[7] Auriol E, Crowder RM, McKendrick R, Rowe R and Knudsen T. Integrating Case-based Reasoning and Hypermedia Documentation: an Application for the Diagnosis of a Welding Robot at Odense Steel Shipyard. Eng. App. of AI. 12 (1999) 691-703.
[8] Brusilovsky P, Cooper DW. Domain, Task, and User Models for an Adaptive Hypermedia Performance Support System. Proc. IUI '02, ACM, San Fransisco, California, USA (2002) 23-30.

[9] Pham DT, Setchi RM. Case-based Generation of Adaptive Product Manuals. Proc. IMechE. B-217 (2003) 313-322.
[10] Pham DT, Dimov SS, Setchi RM, Peat B, Soroka A, Brousseau EB, Huneiti AM, Lagos N, Noyvirt AE, Pasantonopoulos C, Tsaneva DK and Tang Q. Product Lifecycle Management for Performance Support. J. Comp. Inf. Sc. Eng. ASME. 4 (2004) 305-315.
[11] Lagos, N., Setchi, R., Dimov, S.S. Towards the Integration of Performance Support and e-Learning: Context-Aware Product Support Systems. LNCS. 3762 (2005) 1149-1158
[12] Mendes, E, Mosley, N., Watson, I. A comparison of case-based reasoning approaches. Proc. 11th intern. conf. on World Wide Web. Honolulu, Hawaii, USA, (2002) 272-280.