Product Line Instantiation Support for Electronic Shops

Peter Knauber, Michalis Anastasopoulos
Fraunhofer Institute for Experimental Software Engineering (IESE), Sauerwiesen 6
D-67661 Kaiserslautern, Germany
{knauber, anastaso}@iese.fhg.de

Matthias Ress
maxess systemhaus gmbh
Europaallee 3-5
D-67657 Kaiserslautern, Germany
ress@maxess.de

ABSTRACT
People involved in software product line (or domain) engineering development suffer from the lack of tools to support the creation and maintenance of generic product line assets and their instantiation for specific products. Experience from large product line projects like PRAISE and others tells that the support of product line engineering by such tools is very important [1, 4].

This paper shows a solution which supports business process modeling by combining and using existing tools, illustrated with a product line of electronic shops.

1 INTRODUCTION

Probably the most important requirement for applications in the fast-moving market of e-commerce applications is, that they have to be developed fast (that is, in short development cycles) in order to achieve a short time-to-market. The reason for this requirement is simple: the first one entering a new market (segment) with his service is able to establish a new product brand and benefits respectively. The best-known example for this is probably *Amazon.com*: The name Amazon is almost a synonym for Internet bookstores. The value of such a newly established brand can be measured directly in terms of income of the respective company.

For the developers of e-commerce solutions, especially of electronic shops (see Section 2), product line development seems to be a promising approach for several reasons, for example:

- As result from reuse of existing product components, product lines promise shorter development and maintenance time and thus a shorter time-to-market (once the product line infrastructure has been built).
- Most electronic shops provide a very similar functionality combined with a varying look-and-feel. For example, all shops provide means to search their stock but the search functionality can look very different (search for article name, search in article description text, or search in certain article categories). Product lines can support this kind of commonalities among variabilities very well.
- There are hard quality requirements (especially wrt. stability, reliability, and performance) for electronic shops: if there are problems wrt. functionality or performance users are likely to immediately change to another website: “the next service provider is just one mouse click away”.
  Again, software product lines promise a solution: the high amount of reuse to be expected among different shop products derived from the same product line makes it likely to meet the quality requirements mentioned because the reused components have been designed for and (successfully) used in that environment.

The opportunity to develop similar products based on a common core has been recognized by large companies like IBM and Intershop and is reflected in their standard shop-generating products like Websphere [8] and Enfinity [9]. These standard products are a good starting basis for implementing an electronic shop but they have to be extended and customized in order to fulfill customer-specific requirements.

Solutions for the adaptation of modeling tools that focus on the technical level have been shown, for example, an adaptation of Rational Rose for the description of generic reference architectures in the ESAPS project[1] [3]. But these solutions are not applicable in the e-commerce context: The established way to describe requirements for merchandising software (including but not limited to electronic shops) is through definition of the business processes supported by the application, using a notation like *ePK* (see Section 3). This notation is relatively easy to understand for people with non-technical background who are the typical stakeholders in this domain [6] and supported by standard tools like the ARIS Toolset [7]. One advantage of the ARIS Toolset is that its *notation* can easily be extended with symbols for expressing variability in business processes. One drawback of the ARIS solution is that it provides no *semantic support* to handle these variations systematically, for example, take care of dependencies among variations during instantiation of the generic processes towards product- or customer-specific processes.

In this paper we present a solution which supports business process modeling for a product line of electronic shops by combining and using existing tools. Section 2 gives the background of the development organization where the

1. ESAPS (Engineering Software Architectures, Processes and Platforms for System Families) is a large European research project (Eureka Σ! 2023 Programme, ITEA project 99005).

solution has been introduced. Section 3 describes the solution itself and Section 4 summarizes and concludes the paper.

2 Context / Background

Maxess systemhaus gmbh was founded in 1996 with the objective to create a financial system supporting food trade on an object-oriented basis. To this end modern software engineering methods were constantly being taken into account. This financial system provides the foundation for E-Commerce.

A pilot project was launched in 1999 in cooperation with the parent company, which originates from the stationary food trade, with the goal to enable goods ordering over the Internet. The Internet shop was realized with Net.Commerce, Websphere's predecessor. After a short time, a complete warehouse with its own logistics was created providing a high level of comfort to its end users. The success of the first shop, which was used for food procurement at the EXPO2000 at Hannover, Germany, was even greater.

Websphere supports separation between process and layout which is beneficial since the underlying shopping processes remain mostly unchanged whereas the layout changes frequently. Realizing shops for various operators makes the significance of this separation even more apparent. Changes on the processes show up twice in a year to the maximum while the layout may change monthly.

This background motivated the creation of a generic shop. Each shop is an instance of this generic shop. The business processes are derived from a generic process framework and are typically characterized by the absence of some process branches. We use the term feature for process branches that are not necessary for the shop's operation. A shop instance contains the base framework plus additional features. The layout is parameterized and can thus be adapted quickly and dynamically. This brings up some crucial advantages:

- Planning and realization of shops are drastically accelerated because the base framework can be used as-is and already implemented features can be exploited or blocked.
- Effort for individual shop maintenance is reduced since all systems are nearly identical.
- The development of an additional shop feature according to a customer's requirement leads to an overall improvement of the generic framework which may later have positive influence on all customers.

For creating and maintaining the generic shop model a solution was sought which would provide rapid shop instantiation and overview over all features per shop instance.

3 The Solution

Product line genericity can be encountered in various assets: requirements documents, business process diagrams, control flows, source files etc. In the context of this project we have put our initial focus on a framework that deals mainly with the genericity of business process diagrams. Our aim is to extend this framework towards supporting variability in control flows as well as in source code (see Section 4).

For business process modelling maxess systemhaus gmbH uses the ARIS Toolset, a product of IDS Sheer AG [7]. The ARIS Toolset supports tasks related to e-business engineering and business process management.

The main elements of an ARIS business process diagram are business functions, events, process guides and connectors. Figure 1 depicts the notation used for each one of these diagram elements. Business processes described this way are called event-driven process chains (in german: ereignisgesteurte Prozeßketten, in short: ePK)

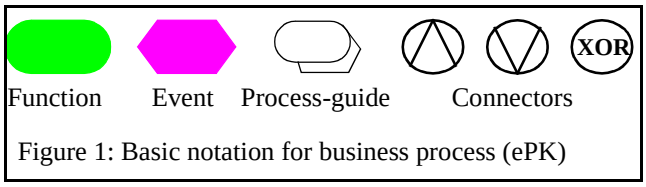


Figure 1: Basic notation for business process (ePK)

Functions transform input to output data, connectors are resolving run-time decisions concerning functions, and events are passive components denoting function completion results. Process guides are links to preceding or succeeding processes [1].

The standard diagram notation provided by ARIS is extensible[1]. We took advantage of this possibility by introducing an additional diagram element which we have called an *optional variant decision*. The idea is to enable the modeling of optional paths in a business process diagram, namely paths that are taken into account only if specific constraints are fulfilled.

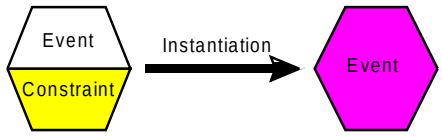


Figure 2: Instantiation of an optional variant decision

The optional variant decision mark is divided in two parts (see Figure 2). The upper (white colored) part represents an event, while the lower (yellow colored) part represents a constraint.

The introduction of such decision marks makes ePK diagrams *generic*. Generic diagrams cover a variety of end-products. The *instantiation* of a generic diagram results in a product-specific diagram. Each product is subject to a set of conditions, which we call constraints, that must be fulfilled in order for the product to be operational. Upon instantiantion constraints are taken into account.

1. Note, that this extensibility supports only the notation, that is, an extension of the semantics underlying the tool is not possible.

For instance, the constraint “Guest login is supported” is fulfilled only in specific electronic shops that support the guest login feature.

During instantiation the affected optional variant decisions are either resolved to ordinary events (see Figure 2) or they get removed from the instantiated diagram. Since optional variant decisions, that are to be removed, are typically starting nodes of optional paths, these paths are completely removed from the instantiated generic diagram.

The following figures should clarify this instantiation mechanism.

3.1 Application Example

Figure 3 contains a generic ePK diagram illustrating a part of the business process for online orders supported by maxess’ electronic shops. This diagram gets instantiated in Figure 4 with only the constraint “Registered Customer Login is supported” being fulfilled.

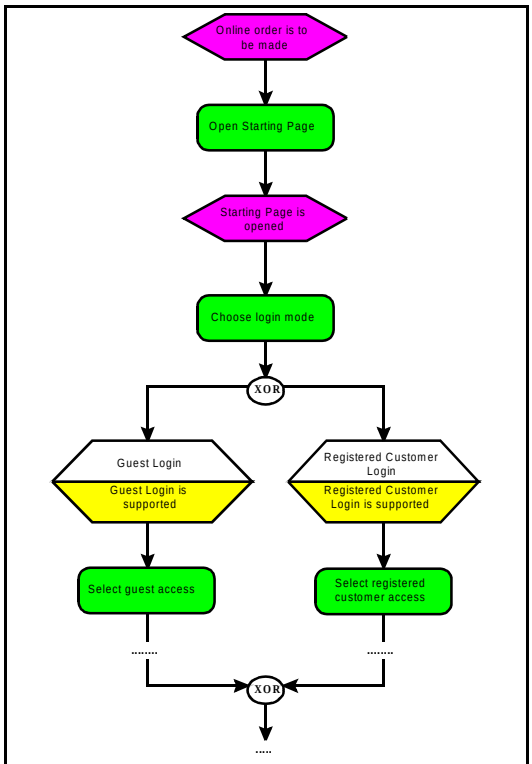


Figure 3: Generic business process diagram

The optional variant decision depending on the fulfillment of the constraint “Guest Login is supported” has been removed in Figure 4 as well as the connector and the corresponding optional path. The “Register Customer Login” optional variant decision became an event and the corresponding path remained available. Since there is only one login mode supported in the instantiated diagram, it makes no sense to keep the XOR connectors after instantiation. Therefore they have been removed.

Generic diagrams have proved to be helpful because they enable centralized business process modeling for a family (or line) of electronic shops. Instances of these generic diagrams must be made available for interaction with customers and for product-specific modelling. Changes on the instances must be coordinated with the generic models. At maxess, changes are made solely on the generic models.

3.1.1 Feature Table

Generic models contain variation points (in our case the optional variant decisions) that are connected with product features. As new features get implemented and new electronic shops are joining the product line, the overview and the control over the dependencies between shops, features, and generic models becomes an important issue.

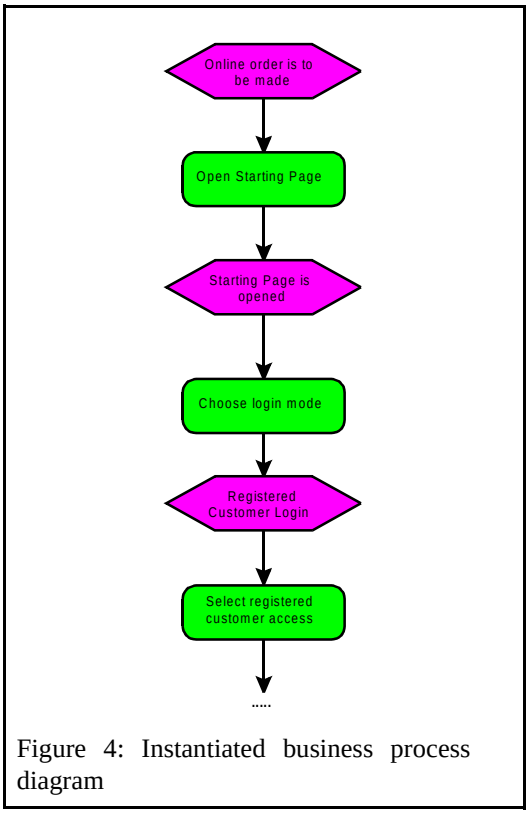


Figure 4: Instantiated business process diagram

Feature tables like the one depicted in Table 1 provide such an overview (the table used in this project can be viewed as a simplification of Product Map used by PuLSE-Eco [5]). Features like “Search Functions” may contain sub-features or feature forms like “Text Search”, “Category Search”, or “Combi Search”. For this reason we have introduced a feature hierarchy which appears in the left most columns of the feature table. In each feature node the leafs, namely the bottom-level sub-features, are the ones that are directly connected with electronic shops (see Table 1). Constraints imposed to each feature node (not to mistake with constraints in the optional variant decisions) appear in the corresponding column.

An attribute of each feature is an entry of the type (n - m : k). The meaning of this notation is the following: If the affected feature is supported by a shop, there are k feature forms out of which a minimum of n and a maximum of m must be supported.

A hierarchy of electronic shops was also required since a shop may have different editions (in our project “small”, “medium”, “large”) which in turn may have different sub-editions (e.g., “with guest access”, “with registered access”). We believe though, that the shop hierarchy must be kept shallow at this point, because a deep hierarchy introduces maintenance and overview problems. This can be done by extracting characteristics of shop editions and by moving them to the feature hierarchy. A more sophisticated edition management is an issue to be addressed in future work.

Features	Constraints	Feature Forms	Shops						Model elements
			Cash & Carry Shop						
			small		middle		large		
			Guest	Reg	Guest	Reg	Guest	Reg	
Supported Search Functions	Text XOR Extended		TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	
(1-3:4)		Categories	no	is	no	is	is	is	101,102
		Extended	no	no	no	no	no	is	121
		Text Search	is	is	is	is	is	no	134
Multiple category search levels	extensible		ok	ok	ok	ok	ok	ok	
(1:4)		2 Levels	is	is	is	is	is	is	156,158,...
		3 Levels	no	no	no	no	must	must	178,...

Table 1: Example of a feature table with one shop’s editions

The cells in the area “Shops” connect shops with features (or feature forms) and constraints. The entries “is”, “no” and “must” mean accordingly that a feature form is, is not or must be supported by an electronic shop. The gray shadowed entries give information about feature constraints and their fulfillment (in the above table “TRUE” means that the logical feature constraint “Text XOR Extended” is met while “ok” means that the number of search levels is extensible as the according constraint requires).

Finally the column “Model elements” provides links to unique identifiers of ePK model elements that relate to each feature form.

At the time being the framework is intended for use by domain engineers. With the existing functionality they are able to change the configuration of a shop in terms of the features it supports and to produce a set of both generic and shop-specific business process diagrams.

At maxess, Microsoft Excel and its macro mechanism is used to represent and extend the above feature table as a spreadsheet (Appendix A: Screenshot of Feature Table shows a screenshot of the partial sheet). The goals of the Excel extension are narrated in the following subsections 3.1.2 and 3.1.3.

3.1.2 Link of feature table with ARIS business process models and instantiation support

For the instantiation of generic models, it was necessary to provide a link mechanism between ARIS and the Excel sheet. This mechanism is depicted in Figure 5.

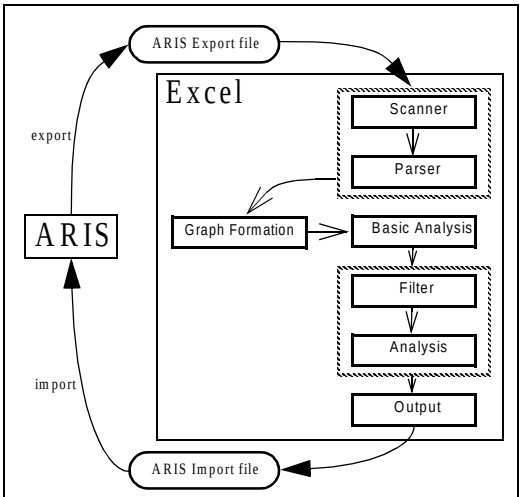


Figure 5: The ARIS-Excel link mechanism

For instantiation, the ARIS text export facility is used: The ARIS toolset is capable of exporting the complete model database in a text file which can be parsed and imported on the Excel side. To start the instantiation process, the sheet offers a user dialog for selecting the shop to be instantiated. Pressing a button causes ARIS to export the text file, starts a filter which parses the file and removes all elements from the generic model that are not part of the instance, writes the reduced model to another text file and provokes the re-import of this reduced model into ARIS.

As Figure 5 shows, the Excel side is also equipped with several analysis components. These components are examining the models initially exported from ARIS and eventually altered through the work with the feature table and make sure that constraints are fulfilled. Inconsistencies before (*Basic Analysis* box) and after the instantiation (*Analysis* box) can be spotted. Modeling guidelines and other predefined rules can be taken into account during the analysis.

3.1.3 Supporting for the creation and management of features and electronic shops

The extended feature table aims at providing user interfaces that enable feature hierarchy management. This firstly includes creating features and sub-features. Secondly users will be in position to declare support of these features by specific electronic shops and to assign constraints. Feature update and the rather unusual feature removal may be furthermore supported.

User interfaces also help adding new electronic shops and shop editions in the product line. New electronic shops can be based on predefined standard shops.

At the time being these user interfaces are at the prototype level.

4 Future Work

This paper describes a way to handle genericity in business process diagrams. Genericity in *control flow* diagrams can be described using optional variant decisions. This supports the direct reuse of existing framework functionality and therefore the realization is considered of low complexity. However it is likely that control flows produced this way cannot effectively lead the implementation. Examining the effects of extending the business process model with control flows and handling the respective constraints is a challenge to be faced in the future versions of our mechanism.

Genericity in code is a difficult issue. It requires parameterized code constructs that can be traced back to control flows and to features. Depending on the programming language used various implementation techniques can be exploited in order to achieve code genericity. Standard environments like IBM Websphere or its predecessor Net.Commerce provide modules, templates, and libraries, that aim at simplifying and speeding up the development process. These ready-made solutions must be parameterized as well which in our project often imposed great difficulties.

Switching from Net.Commerce, which is currently used at maxess systemhaus gmbH, to the Java-based Websphere enhances the possibilities for generic or parameterized programming. In future we intend to examine this new potential in order to support generic *implementation* of electronic shops and their instantiation using our feature table mechanism.

A simple feature table like the one depicted in Table 1 provides information neither about stakeholders and external data involved in the realization of a feature nor about costs and priorities of the respective feature. The extended version contains additional columns providing that knowledge.

It must be noted that at the time being work with the spreadsheet is done manually. The inconsistency management algorithms are partially implemented. Refining these mechanisms, delivering the previously mentioned user interfaces as well as supporting additional columns (e.g., for effort models and stakeholders concerned) have just started.

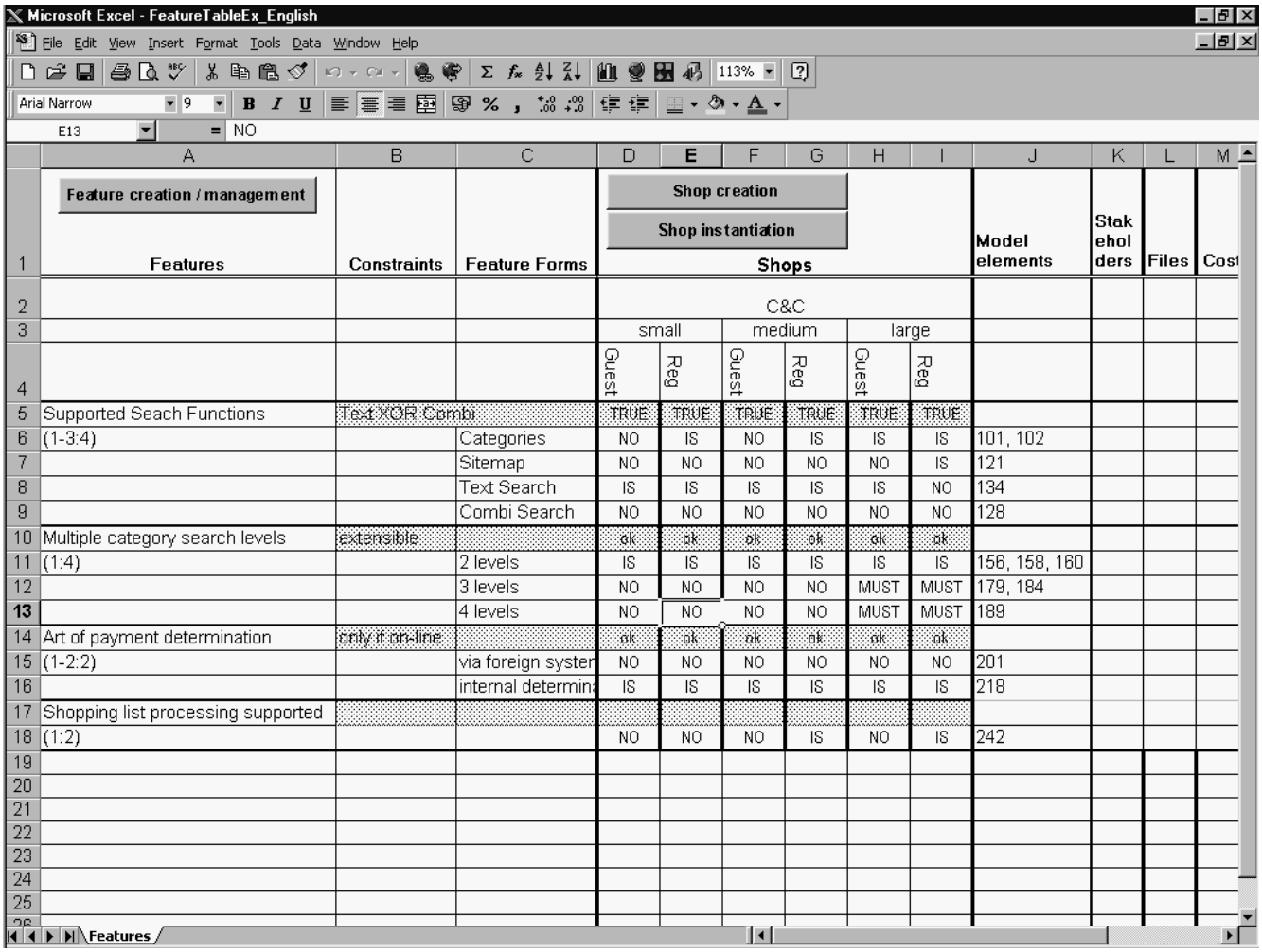
Acknowledgements

We would like to thank all the people who contributed at certain phases to this project. Special thanks go to Ulrike Becker-Kornstaedt and Dr. Cyrus Dethloff who helped with the business process modeling and to Martin Würthner and Gerd Petzold who did the implementation of the ARIS filter.

References

1. Jörg Becker, Reinhard Schütte, Handelsinformationssysteme, Landsberg: Verlag moderne Industrie, 1996
2. William El Kaim, Sophie Cherki, Pascal Josset, Andreas Hein, John MacGregor, Michael Schlick, Renato Vinga-Martins, Ramon Lerchundi: *Tools for Product Line*, PRAISE Report D4.2
3. Oliver Flege: *System Family Architecture Description Using the UML*, IESE-Report No. 092.00/E, can be obtained from http://www.iese.fhg.de/pdf_files/iese-092_00.pdf
4. Peter Knauber, Dirk Muthig, Klaus Schmid, Tanya Widen: *Applying Product Line Concepts in Small- and Medium-Sized Companies*, IEEE Software, September 2000
5. Klaus Schmid: *Scoping Software Product Lines — An Analysis of an Emerging Technology*, Proceedings of Software Product Line Conference (SPLC-1), Denver, CO, 2000
6. Klaus Schmid, Ulrike Becker-Kornstaedt, Peter Knauber, Florian Bernauer: *Introducing a Software Modeling Concept in a Medium-Sized Company*, Proceedings of 22nd International Software Engineering Conference (ICSE 2000), Limerick, 2000
7. Information on the ARIS Toolset can be found at http://www.ids-scheer.de/produkte.htm
8. Information on IBM Websphere can be found at http://www-4.ibm.com/software/webservers/commerce/wcs5_press.html
9. Information on Intershop Enfinity can be found at http://www.intershop.com/products/index.htm?callname=products_enfinity erinnern

APPENDIX A: SCREENSHOT OF FEATURE TABLE



Microsoft Excel - FeatureTableEx_English

File Edit View Insert Format Tools Data Window Help

Arial Narrow 9 B I U

E13 = NO

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Feature creation / management Features	Constraints	Feature Forms	Shop creation Shop instantiation Shops						Model elements	Stakeholders	Files	Cost
2				C&C									
3				small		medium		large					
4				Guest	Reg	Guest	Reg	Guest	Reg				
5	Supported Seach Functions	Text XOR Combi		TRUE	TRUE	TRUE	TRUE	TRUE	TRUE				
6	(1-3:4)		Categories	NO	IS	NO	IS	IS	IS	101, 102			
7			Sitemap	NO	NO	NO	NO	NO	IS	121			
8			Text Search	IS	IS	IS	IS	IS	NO	134			
9			Combi Search	NO	NO	NO	NO	NO	NO	128			
10	Multiple category search levels	extensible		ok	ok	ok	ok	ok	ok				
11	(1:4)		2 levels	IS	IS	IS	IS	IS	IS	156, 158, 160			
12			3 levels	NO	NO	NO	NO	MUST	MUST	179, 184			
13			4 levels	NO	NO	NO	NO	MUST	MUST	189			
14	Art of payment determination	only if on-line		ok	ok	ok	ok	ok	ok				
15	(1-2:2)		via foreign syster	NO	NO	NO	NO	NO	NO	201			
16			internal determina	IS	IS	IS	IS	IS	IS	218			
17	Shopping list processing supported												
18	(1:2)			NO	NO	NO	IS	NO	IS	242			
19													
20													
21													
22													
23													
24													
25													

Features

Figure 6: The extended feature table as an Excel spreadsheet