A Flexible Approach for Generating Product-Specific Documents in Product Lines

Rick Rabiser[1], Wolfgang Heider[1], Christoph Elsner[2], Martin Lehofer[3], Paul Grünbacher[1], and Christa Schwanninger[2]

[1] Christian Doppler Laboratory for Automated Software Engineering, Johannes Kepler University, Linz, Austria
rabiser@ase.jku.at

[2] Siemens Corporate Research & Technologies, Erlangen, Germany
christa.schwanninger@siemens.com

[3] Siemens VAI Metals Technologies, Linz, Austria
martin.lehofer@siemens.com

Abstract. In product line engineering various stakeholders like sales and marketing people, product managers, and technical writers are involved in creating and adapting documents such as offers, contracts, commercial conditions, technical documents, or user manuals. In practice stakeholders often need to adapt these documents manually during product derivation. This adaptation is, however, tedious and error-prone and can easily lead to inconsistencies. Despite some automation there is usually a lack of general concepts and there are "islands of automation" that are hardly integrated. Also, research on product lines has so far often neglected the handling of documents. To address these issues, we developed a flexible approach for automatically generating product-specific documents based on variability models. We applied the approach to two industrial product lines of different maturity using the decision-oriented product line engineering tool suite DOPLER.

Keywords: document generation, model-based approach, product lines.

1 Introduction and Motivation

Software product line engineering (SPLE) [1, 2] traditionally has a strong focus on technical software assets such as architecture or code. Researchers and practitioners are typically adopting feature models [3], decision models [4], UML-based techniques [5], or orthogonal approaches [1] to define the reusable assets' variability. Often these approaches are used to expedite and automate product derivation, e.g., by generating configurations. However, in practice documents play an equally crucial role. Customers, sales people, marketing staff, or product managers frequently work with documents such as offers, user manuals, commercial conditions, or contracts. Also, developers or testers need to provide documents as part of their daily work, e.g., technical documentation or test plans.

Originally published in the Proceedings of the 14th International Software Product Line Conference 2010 (SPLC 2010). The original publication is available at http://www.springerlink.com.

In our collaboration with partners in different domains, we have learned that while the derivation and configuration of products is often at least partly automated, documents are typically still adapted manually. For example, sales people customize offers and user manuals to be delivered to customers and engineers adapt technical documentation for customer-specific development and maintenance. This manual adaptation is often tedious and error-prone and the created documents can easily become inconsistent with the derived software system.

Researchers have presented approaches for extracting reusable assets and variability information from legacy documentation to support defining a product line [2, 6]. There also exists work on developing documentation for software product lines and more generally on reusing and adapting documentation [7, 8]. However, regarding variability modeling and product derivation most research in SPLE has so far focused on technical software assets and treats documents rather as a side issue. Basic support for modeling document variability and adapting documents is for instance available as part of a commercial SPLE tool [9]. Technically, adapting documents seems straightforward. Concepts from model-driven SPLE [10, 11] can for example be applied to structured documents. However, a generic and flexible approach is still missing that provides concepts for automating document generation independently of the concrete type of documents and regardless of the maturity of the product line, i.e., the degree to which variability modeling and generative techniques are already applied.

Based on an existing model-driven SPLE approach [12, 13] we have thus been developing an approach that supports the generation of both deliverable documents and software systems from the same variability model. The approach is based on the observation that many decisions made during product derivation are relevant to both types of assets. For example, the decision to include a certain feature will also require the inclusion of the user documentation of this feature. Decisions made in product derivation typically also have an effect on sales documents such as offers or contracts. For example, descriptions of selected features must also be part of such documents.

Using the tool suite DOPLER [12, 13] we successfully applied our approach in the domain of industrial automation to two product lines of different maturity:

The first application example is a mature software product line for the process automation of continuous casting machines in steel plants from Siemens VAI. The architecture of this software is modularized and built as Java Beans using the Spring component framework [14]. In this case the variability has already been modeled for the technical software assets [12, 15] and configuration files for concrete solutions can be generated using DOPLER. Examples for variability in a continuous caster delivered by Siemens VAI are the number of strands, the choice of whether to include a cooling system, and the type of cooling. Such variability not only affects technical software assets but also directly shapes the user documentation of the software. Interviews with domain experts revealed that the manual adaptation of documentation for each customer can be tedious and automation would be highly beneficial.

The second industrial application example is about automating the generation of sales documents like customer-specific offers, product descriptions, and commercial condition documents at Siemens AG for a family of electrode control systems for electric arc furnaces (EAF) in steel production. The corresponding sales department has to provide many specific documents to prospective customers. Creating these documents requires the sales people to “parse” and “process” the documents manually

each time a quote has to be submitted. Erroneous offers may result in actual losses or legal issues; therefore the manually created documents are thoroughly reviewed and checked for quality. This altogether can extend the duration for creating offers to several weeks. Currently the offer process takes two weeks in average and 120 offers are written a year. The time between the initial customer contact and the delivery of the offer matters a lot to customers. The goal is thus to reduce the time needed for an offer to at most one week to improve customer satisfaction. Fortunately, in this industrial case the variability in the documents has already been known to a large extent. Many of the documents already contained explicit variability information about product features, target environment, and commercial and legal conditions.

Our experience shows that in the first case of a mature software product line it is possible to largely reuse the existing variability models for generating documents. In our second application example automated product derivation techniques have not yet been applied. In this case, however, document generation provides a convincing showcase for the benefits of variability analysis, modeling, and automation. In both projects decision models were considered very helpful for describing variability, especially by non-technical stakeholders.

The remainder of this paper is structured as follows: We first give an overview of our flexible document generation approach. We then present the concrete technical realization based on DOPLER. We illustrate the feasibility of the approach by providing details of applying it in the two industrial application examples. We discuss related work and conclude the paper with an outlook on future work.

2 Approach

Our approach for document generation in product derivation is independent of the maturity of the product line, the concrete variability modeling technique, and the type of documents to be generated. For instance, model-driven SPLE techniques [10, 11] could be used for implementation. The approach comprises four steps which are usually conducted iteratively:

(1) *Elicit and analyze variability in documents*. An industrial product line rarely starts from scratch, in particular when considering automation. Examining existing documents such as user manuals or contracts helps to reveal variability. Therefore a product line expert familiar with variability modeling needs to analyze such existing assets. Additionally, it is advisable to conduct workshops with domain experts from product management, sales, and development. Such experts have frequently been dealing with variability in documents in the past and they know which manual adaptations have been most relevant [6]. The document analysis shows what can vary (variation points) and how it may vary (variants). The key challenge is to find the right level of detail and granularity. It does not make sense to elicit and analyze all possible variability. Instead, domain experts should focus on the most relevant variability that can potentially attain the highest cost savings.

(2) *Create or adapt variability models*. The product line expert uses the collected information to either adapt existing models or to create new ones. If existing models already guide and automate product derivation, a considerable part of the models

might also be used for automating document generation. For instance, the selection of a particular feature in product derivation might not only include a software component, but also include related documents. There are no restrictions regarding the technique used to model document variability. However, the technique should be flexible to allow more advanced automation during product derivation in the future.

(3) *Choose or develop a variability mechanism and a corresponding generator for domain-specific document formats*. A generator is required that automates the creation of product-specific documents according to variant selections. Such a generator typically relies on explicitly defined variation points in the documents, whereas the mechanisms used to define them depend on the documents' formats. However, some unstructured formats (e.g., txt, rtf) are not well suited for that purpose. Therefore, documents need to be converted to formats for which document processors either exist [16] or can be developed with reasonable effort (e.g., document formats of popular office suites like Microsoft Word can be extended using markups [9]). Beyond a variability markup mechanism it is useful to partition large documents and to add meta-information for coping with coarse-grained variability. Many existing document assembly tools use XML-based formats for that purpose.

(4) *Augment the documents with variability information*. Product line experts formally specify the variation points and variants from step (1) in the documents and relate them to the variability models from step (2) using the mechanism chosen in step (3). During product derivation the generator resolves variation points according to a particular input configuration and produces product-specific documents.

3 Tool Architecture and Realization

We implemented the described approach using existing technologies and tool suites. For variability modeling we use the decision-oriented DOPLER approach [12, 15]. For defining variability in documents we adopted and extended the DocBook system [16] and developed a generator extension for DOPLER [13].

3.1 The DOPLER Approach and Tooling

DOPLER is a decision-oriented SPLE approach [12, 15] comprising a variability modeling tool and a configuration wizard to present variability to users in product derivation. DOPLER variability models contain Assets and Decisions (cf. upper part of Fig. 1). *Assets* represent the core product line artifacts (e.g., technical components or documents). Assets can depend on each other functionally (e.g., one component requires another component) or structurally (e.g., a paragraph is part of a chapter). DOPLER allows modeling assets at arbitrary granularity and with domain-specific attributes and dependencies, based on a given set of basic types. Users can create domain-specific meta-models to define their asset types, attributes, and dependencies.

In DOPLER variation points are defined and presented as *decisions*. Important attributes of decisions are a unique id, a question that is asked to a user during product derivation, and a decision type (Boolean, enumeration, string, or number). Decisions

can depend on each other hierarchically (if a decision needs to be made before another decision) or logically (if making a decision changes the value of another decision). The decision type describes the range of the possible answers and can be further constrained with validity conditions. Decision models have proven useful in application engineering as they allow describing variability at a higher level of abstraction matching the problem space terminology of domain experts [17]. Decision models in conjunction with asset models also increase flexibility, as a single decision model can be used with several asset models defining different types of assets such as components or documents. In DOPLER, assets are linked to decisions via inclusion conditions defining when a particular asset is included in a derived product. Asset attributes can also be dependent on answers to decisions to enable the customization of assets.

3.2 Modeling Documents and their Variability with DOPLER

DOPLER allows defining domain-specific asset types by creating a meta-model for a particular organization or context. We can thus include documents or parts of documents as dedicated assets in our models. We have defined the generic asset type `document fragment` representing arbitrary parts of documents, such as chapters or sections. Each `document fragment` has an attribute `location` (of type URL) in addition to the default model attributes id and description. We also defined that `document fragments` can be part of other fragments, e.g., a section might be part of a chapter, and that they can contribute to other assets, e.g., if a fragment describes a particular component. The generic asset type `document fragment` can be further refined with additional attributes and relations for domain-specific purposes. Fig. 1 shows the DOPLER meta-model extended with `document fragments`.

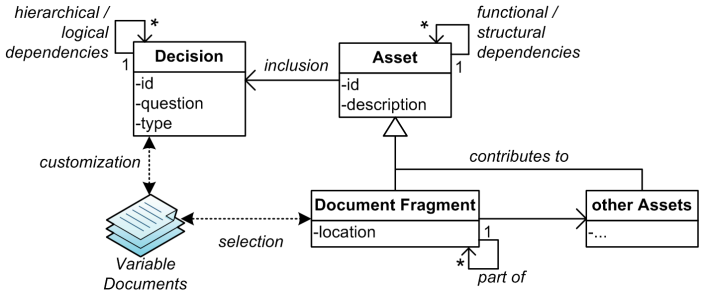

Fig. 1. Extended DOPLER meta-model with documents.

Decisions represent document variability as questions that a user is expected to answer during product derivation. `Document fragment` assets represent arbitrary parts of documents and are used to model coarse-grained variability. Explicitly representing parts of documents as assets in a model provides an additional level of abstraction for defining document variability and for selecting a particular chapter or section for inclusion in the derived documents. It would be possible to describe document variability at the level of decisions only (and using the values of decisions to customize documents *and* select parts of documents). However, we have learned that the addi-

tional level of abstraction helps domain experts in understanding coarse-grained document variability. Also, the same decision often can be used for several *document fragments* or for including both documents and components thus making the approach more flexible. For example, the *document fragment coolingchapter* represents the documentation of the cooling system and depends on the value of the decision "*Shall the cooling system be delivered?*". As soon as a user answers the question with *yes*, the asset *coolingchapter* is marked to be included in the derived product to resolve this coarse-grained variability. To support fine-grained customizations, we developed a variability mechanism based on DocBook.

3.3 A Variability Extension for DocBook

DocBook [16] is a collection of standards and tools for technical publishing proposed by a consortium of software companies as a computer documentation standard. The core of DocBook is a Document Type Definition (DTD) defining content elements and their possible relations that authors can use for creating documents. For example, the *book* element can contain a *title* element which can contain *para* and *chapter* elements. Using the DTD and XML syntax, authors can write text content with markup using arbitrary XML tools. Components for editing, typesetting, and version control can be combined as needed and a set of XSL style sheets is available for converting content to arbitrary target formats such as HTML or PDF. DocBook is well-suited for automatically processing documents (see e.g., [16, 18]). It can be compared with LaTeX but uses XML markup.

DocBook provides a profiling mechanism for extending the DTD by defining new elements and attributes. We use this mechanism to define elements and attributes for implementing variation points in documents. For instance, we defined an attribute *doplerdoc* that can be used with all elements in DocBook files. The value of the *doplerdoc* attribute can refer to the unique id of a decision or an asset (i.e., *document fragment* asset). Adding the *doplerdoc* attribute, e.g., to a *chapter* or *para* element, allows tagging it as optional/alternative depending on whether a particular asset is included in a derived product. The element *doplerdocplaceholder* can be used in combination with the *doplerdoc* attribute to define a placeholder in the text that will be filled with the answer to a particular decision (cf. Listing 1).

```
<!--chapter "cooling" is included in the documentation of the product if document fragment
asset "coolingchapter" is included due to some decision made in product derivation.-->
<chapter id="cooling" doplerdoc="coolingchapter">
      <!-- paragraph "cooling_mechanism" includes text with a placeholder. This placeholder
      is replaced with the answer set to decision "cooling_mech" in product derivation. -->
      <para id="cooling_mechanism">
      For secondary cooling, the caster supports the
      <doplerdocplaceholder doplerdoc="cooling_mech"/> mechanism. ...
      </para>
      <para id="...">...</para>
</chapter>
```

Listing 1. Example showing how DocBook files can be parameterized to make them variable and how they can be related to a DOPLER variability model.

We analyzed the technical user documentation as well as sales documents (i.e., offers and commercial conditions) in the domain of industrial automation systems to identify common patterns of variability in documents. Table 1 presents the most common and most relevant types we found, provides examples, and demonstrates how the different types of variability can be realized using DocBook and DOPLER.

Table 1. Implementing document variability using DocBook and DOPLER.

Document Variability	**Description and Example**	**Implementation in DocBook**

Placeholders	Items that are replaced with text or values that are entered and calculated during product derivation, e.g.,
	- customer details (name, company, address, etc.),
	- total price,
	- return on investment values.
		Use designated *doplerdocplaceholder* element and use a *doplerdoc* attribute to relate the placeholder with a decision, e.g.,
		`<section id="caster">`
		`The caster has`
		`<doplerdocplaceholder`
		`doplerdoc="numStrands"/>`
		`strands. </section>`

Optional text	Chapters or sections, paragraphs, sentences, or even words or letters are added or removed depending on the features to be delivered, e.g., chapters in a user manual or in a bidding document.	Mark DocBook element as optional using the *doplerdoc* attribute to relate it with a *document fragment* asset, e.g.,
		`<chapter id="hmi"`
		`doplerdoc="hmichapt">`
		`... </chapter>`

Alternative text	Chapters or sections, paragraphs, sentences, or even words or letters that alternate depending on the target market, customers and system environment, e.g.,
	- country-specific commercial conditions and policies,
	- different operating systems,
	- units (e.g., metric vs. imperial system).
		Enclose alternative parts with DocBook elements and add *doplerdoc* attributes related with *document fragments*, e.g.,
		`<section id="dex"`
		`doplerdoc="dexchapter">`
		`Data Exchange is supported via`
		`<phrase doplerdoc="asciiphrase">`
		`ASCII</phrase>`
		`<phrase doplerdoc="dbphrase">`
		`DataBase</phrase>`
		`<phrase doplerdoc="tcpipphrase">`
		`TCP/IP</phrase>`
		`in your system. ...</section>`

Cross references	References to document internals and links to external documents not included in the generated document must be found and replaced to avoid tangling references, e.g.,
	- the main index,
	- figure and table indices,
	- references to docs like country-specific legal documents.
		References and links in DocBook must be related to the same *document fragment* assets as the parts they reference, e.g.,
		`<xref linkend="hmi"`
		`doplerdoc="hmichapt"/>.`
		
		If additional text describes the link (e.g., "...refer to chapter..."), this text must also be enclosed with an XML element dependent on the *hmichapt* *document fragment* asset.

Simple grammatical variability	Mainly regards singular vs. plural but also gender, e.g.,
	- 1 strand vs. 2 or more strands,
	- multiple drives vs. the drive.
		Respective text has to be enclosed with elements and marked with the `doplerdoc` attribute. An additional dependency to a numerical decision allows defining whether to use singular or plural, e.g.,
		`The caster has`
		`<doplerdocplaceholder`
		`doplerdoc="numStrands"/>`
		`strand<phrase`
		`doplerdoc="numStrands#2+">`
		`s</phrase>.`

Media objects	Media objects (e.g., images) might have to be replaced depending on sales aspects and the delivered system, e.g.,
	- customer's logo,
	- user interface.
		Mark DocBook element as optional using designated `doplerdoc` attribute, e.g.,
		`<mediaobject`
		`doplerdoc="hmichapt">`
		`...`
		`</mediaobject>`

Formatting/ Layout	Different styles might be required/desired for different users, e.g., - A4 vs letter size, - different color schema.	Formatting/Layout is achieved using XSL transformation and CSS style sheets. The selection which XSL file and/or CSS style sheet is used can be related to a property file which can be generated based on decisions.

3.4 Generating Documents using DocBook

Fig. 2 depicts an overview of our tool architecture. The DOPLER tool suite supports defining document variability in models, augmenting documents with variability information, as well as communicating document variability to end-users using a configuration wizard (cf. Fig. 3 in Section 4.2). Traceability from the model's assets and decisions to DocBook is achieved via the dedicated XML attribute `doplerdoc` and XML element `doplerdocplaceholder`.

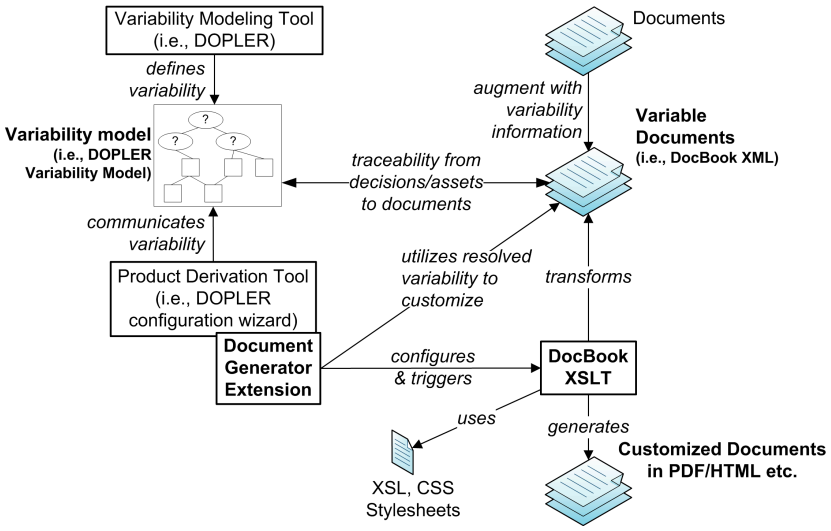

Fig. 2. Architecture for managing and customizing documents in a product line.

The DOPLER product derivation tool (configuration wizard) can be extended with domain-specific generators. We developed a generator that uses the answers to decisions and the selected `document fragment` assets to compose documents. The generator customizes the documents using decision values and post-processes the documents using DocBook's XSLT engine and style sheets (which themselves can depend on decision values). Different output formats can be generated (e.g., PDF or HTML) according to the selected XSLT transformations and style sheets.

4 Industrial Application Examples

We applied the described approach in two industrial product lines with different maturity. In a project with Siemens VAI variability models describing a continuous casting automation software product line were already available and generating product configurations was already supported. The goal in this project was to also support generating technical user documentation. In a second project with Siemens AG, we applied the document generation approach to the EAF product line where no variability modeling has been used before. The goal of this second project was to support the automated generation of customer-specific offers, product descriptions, and commercial condition documents.

4.1 Industrial Application Example I: Generating Technical User Documentation for a Continuous Casting Automation Software Product Line

In the project with Siemens VAI we applied our approach with the goal of automating the generation of customer-specific technical user documentation by reusing the decisions modeled for software configuration. Parts of the technical user documentation were already available as DocBook sources which made adding variability meta-information pretty straightforward (cf. Section 3.3).

(1) *Elicit and analyze variability in documents*. Based on interviews with domain experts, we identified relevant variability in the documentation (cf. Table 1). According to the domain experts, the following variation points occur frequently when adapting the documentation for a particular customer: (i) parts of the documentation are optional depending on the parts of the system to be delivered, (ii) cross references have to be adapted to avoid tangling references when particular sections are not shipped (the same applies to indices), (iii) numerous placeholders in the text need to be replaced (e.g., customer name), (iv) grammatical changes have to be performed (e.g., strand vs. strands), and (v) specific documents need to be deployed if customers intend to develop extensions to the system instead of using it "out of the box".

(2) *Create or adapt variability models*. We extended the generic DOPLER metamodel for modeling `document fragment` assets as discussed in Section 3.2. We captured the diverse parts of the documentation as `document fragment` assets and

related these assets to the decisions. In about 70% of the cases we just used software configuration decisions already defined in the existing variability models. In the remaining cases we added new decisions specifically for document generation.

(3) *Choose or develop a variability mechanism and a corresponding generator for the domain-specific document format*. We used the generator extension for the DOPLER configuration wizard that reassembles the technical user documentation according to the decisions for the derived product (cf. Section 3.4). This generator resolves the variability and uses libraries for performing the proper transformations from the DocBook sources into the selected target format (e.g., HTML or PDF).

(4) *Augment the documents with variability information*. We augmented the existing DocBook sources with the elicited variability, i.e., for every kind of variability mentioned by the domain experts, we implemented at least one example to demonstrate the feasibility of our approach (cf. Table 1).

The resulting tool chain supports modeling the variability of the software and the documents in an integrated manner and automates the generation of both product configurations and technical user documentation using a single decision model.

4.2 Industrial Application Example II: Generating Sales Documents for an Electrode Control System for Electric Arc Furnaces (EAF)

In the EAF project we applied our approach to support the automated generation of customer-specific sales documents. When using the DOPLER configuration wizard variability can be resolved in interactive interviews. Prospective customers and sales people jointly answer the questions defined in the decision model (cf. Fig. 3). Based on their input several documents are generated automatically, e.g., offers, product descriptions, return-on-invest (ROI) estimations, and commercial conditions.

(1) *Elicit and analyze variability in documents*. Together with domain experts, we analyzed existing Microsoft Word documents for offers, price lists, product descriptions, and commercial conditions to identify commonalities and variability. According to the domain experts, the following variability occurs frequently when adapting the documents for a particular customer: (i) *in offers* the customer details, price lists, total prices, and ROI values differ depending on the customer and selected features (i.e., "sellable units"); (ii) *in product descriptions* the component description chapters differ depending on the selected features of the product; (iii) *in commercial conditions* the values and text with regard to the payment method, paying installments, delivery details, warranty conditions, currency, and validity of the offer can differ. Many variation points were obvious (e.g., price list items) and some variability was already tagged with comments within the Word documents. Some variation points span more than just one document. For example, at customization time, the offer is extended with the offered items of the price list and values for the total price and ROI are calculated. The product description document is extended with additional chapters for every additional system component.

(2) *Create or adapt variability models*. We slightly adapted the DOPLER metamodel for dealing with sales documents (cf. Fig. 1 in Section 3.2). The parts of documents defined in the model represent sellable units. Thus, we called the asset type representing document fragments `feature` in this case (not to be confused with the

notion of a feature in feature modeling!). `Feature` assets have the attribute *`price`*. This is required for the generator which in this case calculates values like the total price and the ROI besides generating documents. We modeled decisions for capturing customer details (e.g., name and address), the system environment (e.g., voltage of available power supply), and values needed for ROI calculations (e.g., price for electricity). In total we extracted 101 decisions for deriving the complete set of documents with all variants for prospective customers. These decisions are hierarchically organized and numerous interdependencies are defined.

(3) *Choose or develop a variability mechanism and a corresponding generator for the domain-specific document format*. We reused the generator extension for the DOPLER tool developed for Siemens VAI and extended it for generating offers. For business values like the total price and ROI the generator performs the necessary calculations and saves the results to text files. The content of these files is referenced in the DocBook sources to include these values in the generated documents.

(4) *Augment the documents with variability information*. Like for Siemens VAI we implemented the variation points in DocBook sources we created manually from the existing Word documents. While we could also have augmented the Word documents directly (e.g., using markups [9]), our industry partners decided to use DocBook. Within the DocBook source files we applied the described document variability concepts (cf. Table 1), such as placeholders, optional parts, alternatives, and cross references to enable the automated generation of sales documents.

Fig. 3 shows the DOPLER configuration wizard generating offers for the Electrode Control System for EAF.

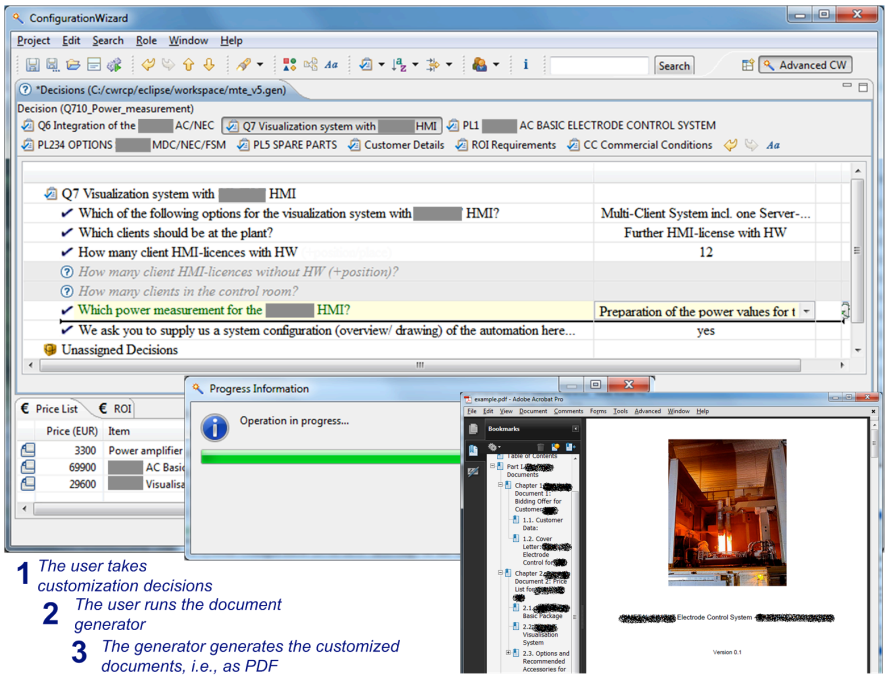

Fig. 3. Making decisions in the DOPLER derivation tool and generating sales documents in the EAF example (partly blurred due to non-disclosure agreements).

The user can make customization decisions by answering questions. The document generator can be started at any time. Based on the user's answers to questions the tool computes the required list of document assets. The generator then creates customized documents in PDF format based on this list.

5 Related Work

We present related work grouped in two areas of research, i.e., research on model-driven product derivation and research on modeling and generating documents in a product line context.

Model-driven product derivation. Several approaches to model-driven product derivation in SPLE have been proposed using a variety of concepts and technologies to support model transformations and product generation. For example, Groher and Völter [10] integrate concepts from model-driven and aspect-oriented software development to support model composition and transformation in an SPLE context. Another example is the work by Ziadi and Jézéquel [11], who present an approach using UML model-transformation to support product derivation in the context of SPLE with UML. Sánchez *et al*. [19] propose VML4*, a tool-supported process for specifying product derivation processes in SPLE that automates product derivation with model transformations. Similar tools have been proposed, e.g., FeatureMapper [20] or CVL [21]. These and other existing model-driven product derivation approaches and tools however have so far focused on technical software assets, mainly software architecture and treat documents as a side issue or not at all. We think many of the existing model-driven approaches and tools could also be extended to support modeling document variability and document generation, similar to what we did in this paper with DOPLER. However, the approaches are not well suited to be used by sales experts and other non-technical stakeholders.

Modeling and generating documents in a product line context. Nicolás and Toval [22] present a systematic literature review on the generation of textual requirements specifications from models in software engineering. The review shows that a lot of work exists on generating requirements specifications from models of different kinds but that there is a lack of support for modeling and generating documents of different types in SPLE. Dordowsky and Hipp [8] report on the introduction of SPLE for an avionics system where they also had to consider documents (i.e., software design and interface design descriptions) and their generation. Their domain-specific solution however does not seem to be generally applicable for modeling and generating documents in SPLE. Koznov and Romanovsky [7] propose DocLine, a method for developing documentation for software product lines that allows reusing document fragments. They also propose a "documentation reuse language" for defining documentation. They focus on documentation for the product line and derived products but do not specifically address generating arbitrary other kinds of documents like offers.

Gears [23] includes document adaptation in an approach manipulating all kinds of product line artifacts. The company pure-systems describes how to use its commercial variability management tool pure::variants to customize Microsoft Word documents depending on features selected in a feature model [9]. With our approach, we do not focus on one specific format of documents (DocBook could be replaced quite easily by developing a new generator) and suggest to explicitly define coarse-grained document variability in models (*document fragment* assets). Most importantly, we use end-user customization tools based on decision models to resolve variability.

6 Conclusions and Future Work

The main contributions of this paper are: (1) a flexible, tool-supported approach to model the variability of documents in SPLE and end-user support for generating product-specific documents in product derivation and (2) an initial evaluation based on two industrial examples where both technical as well as business documents are important. We implemented the approach using the DOPLER and DocBook tool suites. However, a different variability modeling approach and document format might be used for the same purpose. Especially if converting the source documents to DocBook is not feasible in an organization – due to effort or internal regulations – other formats have to be extended to express variability, e.g. like in [9]. The focus is (and has to be) on flexibility with regard to possible document types and different types of variability in documents. Our extensible tool support allows development and integration of arbitrary generators for generating documents based on a variability model. Moreover, the questionnaire-like structure of DOPLER decision models supports stakeholders without technical background knowledge.

Both industrial partners plan to integrate our approach in their process and tool landscape. We will support pilot projects to further improve our approach and tools. The extensibility of our tools will enable developing extensions for the integration of our tools in the organizations' tool environment. For instance, one of our partners uses a proprietary tool for generating technical specifications that could substitute DocBook. Furthermore existing document management systems and collaboration infrastructure will be used for managing standard text that is subject to changes regularly, e.g., for legal conditions. This will prevent errors in the offer process and helps keeping review cycles short. In the EAF example the results will be used as a first step to further automate product derivation. For example, the layout of the Electrode Control System and the software configuration might be generated from the same decision model.

We plan to perform further case studies in other domains to validate our approach and identify useful process automations, e.g., the automatic conversion of Microsoft Word documents to DocBook. We refer to unique id's of model elements in documents which can complicate evolution of models and documents. We have been developing incremental consistency checking support for code assets [24] and plan to adapt it for documents in our future work. Furthermore, we want to study more complex settings like multi-stage configuration. Sales people, technicians, and lawyers have to work together to get all the decisions right [25]. Similarly the collaboration of

a headquarter with regional organizations complicates matters. Some decisions might depend on the region, while others are to be decided by the headquarter.

Acknowledgments. This work has been supported by the Christian Doppler Forschungsgesellschaft, Austria and Siemens VAI Metals Technologies as well as Siemens Corporate Research & Technologies. We thank Manuel Wallnöfer who contributed to the development of a document generator for one industrial application.

References

1. Pohl, K., Böckle, G., van der Linden, F.: Software Product Line Engineering: Foundations, Principles, and Techniques. Springer (2005)
2. van der Linden, F., Schmid, K., Rommes, E.: Software Product Lines in Action − The Best Industrial Practice in Product Line Engineering. Springer Berlin Heidelberg (2007)
3. Kang, K. C., Lee, J., Donohoe, P.: Feature-Oriented Product Line Engineering. IEEE Software, vol. 19(4), IEEE Computer Society (2002) 58–65
4. Schmid, K., John, I.: A Customizable Approach to Full-Life Cycle Variability Management. Journal of the Science of Computer Programming, Special Issue on Variability Management, vol. 53(3), Elsevier (2004) 259–284
5. Gomaa, H.: Designing Software Product Lines with UML. Addison-Wesley (2005)
6. Rabiser, R., Dhungana, D., Grünbacher, P., Burgstaller, B.: Value-Based Elicitation of Product Line Variability: An Experience Report. In: Heymans, P., Kang, K. C., Metzger, A., Pohl, K. (eds.): Proceedings of the Second International Workshop on Variability Modelling of Software-intensive Systems (VaMoS 2008), Essen, Germany, University of Duisburg Essen (2008) 73–79
7. Koznov, D. V., Romanovsky, K. Y.: DocLine: A method for software product lines documentation development. Programming and Computer Software, vol. 34(4), Springer Science+Business Media LLC. (2008) 216–224
8. Dordowsky F., Hipp, W.: Adopting Software Product Line Principles to Manage Software Variants in a Complex Avionics System. In: McGregor, J. D. (ed.): Proceedings of the 13th International Software Product Line Conference (SPLC 2009), San Francisco, CA, USA, ACM International Conference Proceeding Series; Vol. 446, CarnegieMellon University (2009) 265–274.
9. Pure:systems GmbH: Automatic Generation of Word Document Variants. http://www.pure-systems.com/flash/pv-word-integration/flash.html (2010)
10. Voelter, M., Groher, I.: Product Line Implementation using Aspect-Oriented and Model-Driven Software Development. In: Proceedings of the 11th International Software Product Line Conference (SPLC 2007), Kyoto, Japan, IEEE Computer Society (2007) 233–242
11. Ziadi T., Jézéquel, J. M.: Software Product Line Engineering with the UML: Deriving Products. In: Käkölä, T., Duenas, J. (eds.): Software Product Lines. Springer (2006) 557–588
12. Dhungana, D., Grünbacher, P, Rabiser, R., Neumayer, T.: Structuring the Modeling Space and Supporting Evolution in Software Product Line Engineering. Journal of Systems and Software (2010) in press, doi:10.1016/j.jss.2010.02.018
13. Dhungana, D., Rabiser, R., Grünbacher, P., Lehner, K., Federspiel, C.: DOPLER: An Adaptable Tool Suite for Product Line Engineering. In: Proceedings of the 11th International Software Product Line Conference (SPLC 2007), vol. 2, Kyoto, Japan, Kindai Kagaku Sha Co. Ltd. (2007) 151–152

14. Johnson, R., Höller, J., Arendsen, A.: Professional Java Development with the Spring Framework. Wiley Publishing (2005)
15. Rabiser, R., Grünbacher, P., Dhungana, D.: Supporting Product Derivation by Adapting and Augmenting Variability Models. In: Proceedings of the 11th International Software Product Line Conference (SPLC 2007), Kyoto, Japan, IEEE Computer Society (2007) 141–150
16. Stayton, B.: DocBook XSL. Sagehill Enterprises (2005)
17. Grünbacher, P., Rabiser, R., Dhungana, D., Lehofer, M.: Model-based Customization and Deployment of Eclipse-Based Tools: Industrial Experiences. In: Proceedings of the 24th IEEE/ACM International Conference on Automated Software Engineering (ASE 2009), Auckland, New Zealand, IEEE/ACM (2009) 247–256.
18. Walsh, N., Muellner, L.: DocBook: the definitive guide. O'Reilly (1999)
19. Sánchez, P., Loughran, N., Fuentes, L., Garcia, A.: Engineering Languages for Specifying Product-Derivation Processes in Software Product Lines. In: Gašević D., Lämmel, R., Wyk, E. (eds.): Proceedings of the First International Conference on Software Language Engineering (SLE 2008), Toulouse, France, Springer (2008) 188–207
20. Heidenreich, F., Kopcsek, J., Wende, C.: FeatureMapper: mapping features to models. In: Proceedings of the 30th International Conference on Software Engineering, Leipzig, Germany, ACM (2008) 943–944
21. Haugen, Ø., Møller-Pedersen, B., Oldevik, J., Olsen, G., Svendsen, A.: Adding Standardized Variability to Domain Specific Languages. In: Geppert, B., Pohl, K. (eds.): Proceedings of the 12th International Software Product Line Conference (SPLC 2008), Limerick, Ireland, IEEE Computer Society (2008) 139–148
22. Nicolás, J., Toval, A.: On the generation of requirements specifications from software engineering models: A systematic literature review. Information and Software Technology, vol. 51(9), Elsevier (2009) 1291–1307
23. Krueger, C.: The BigLever Software Gears Unified Software Product Line Engineering Framework. In: Geppert, B., Pohl, K. (eds.): Proceedings of the 12th International Software Product Line Conference (SPLC 2008), vol. 2, Limerick, Ireland, Lero (2008) 353
24. Vierhauser, M., Dhungana, D., Heider, W., Rabiser, R., Egyed, A.: Tool Support for Incremental Consistency Checking on Variability Models. In: Benavides, D., Batory, D., Grünbacher, P. (eds.): Proceedings of the 4th International Workshop on Variability Modelling of Software-intensive Systems (VaMoS 2010), Linz, Austria, ICB-Research Report No. 37, University of Duisburg Essen (2010) 171–174
25. O'Leary, P., Rabiser, R., Richardson, I., Thiel, S.: Important Issues and Key Activities in Product Derivation: Experiences from Two Independent Research Projects. In: McGregor, J. D. (ed.): Proceedings of the 13th International Software Product Line Conference (SPLC 2009), San Francisco, CA, USA, ACM International Conference Proceeding Series; Vol. 446, CarnegieMellon University (2009) 121–130