ABSTRACT
A comprehensive software development process needs some adjustment before it can be used: It needs to be tailored to the particular organization’s and project’s setting. The definition of an appropriate tailoring model is a critical task. Process users need tailoring that enables them to trim the process regarding the actual needs. Process engineers need a method and a tool to define a valid model. The SE Book of T-Systems contains a feature model to describe variable parts of the process model and relations and constraints between these parts. The notation and semantics of feature models can be used to visually author a consistent and valid tailoring model. In this paper we present a tool for visual modeling and validation of process model tailoring based on feature models using the SE Book of T-Systems as an example. The tool is based on a domain-specific language that represents the process model. It leverages the semantics of feature models to provide an easy-to-use editor for tailoring-enabled process models.

Categories and Subject Descriptors
D.2.2 [Design Tools and Techniques]: Programmer workbench; D.2.6 [Programming Environments]: Graphical environments, Integrated environments

Keywords
development process, feature-model, visual modeling

1. INTRODUCTION
Software development process models summarize and formalize experience, knowledge, and best practices of successful projects. Standard processes such as the Rational Unified Process (RUP, [7]) are designed as a superset of development processes to be applicable in many different software development projects. Before such a generic process model can be used, it has to be tailored to reflect the circumstances of the concrete project. As a process model is of limited value without project-specific customization, a tailoring method can be considered a critical part of the process itself.

The so-called SE Book is the in-house development process of T-Systems International GmbH1. It uses a feature model known from Software Product Line engineering [4] to describe the mandatory and the variable parts of the process model. The process user can customize the process to the project’s circumstances using a tool that makes use of the feature model.

1.1 Problem Statement
Similar to RUP or the V-Modell XT, the SE Book is very large. It consists of a couple of thousand elements with relations between them. Virtually every one of those elements can be subject to tailoring, meaning that it is affected by the selection or deselection of a feature in the feature-model. To avoid configurations where two features would have conflicting effects, the process engineer therefore needs to have in mind potentially the whole model during design. The process engineer could be relieved from dealing with some of this complexity by appropriate tool support.

1.2 Contribution & Context
In this paper we present a tool that supports the process engineer during the design of a feature model for tailoring. The tool is based on the platform PDE [8, 17], which supports process engineers during the design- and authoring phases of process model development and maintenance. PDE is a generic platform. We show how the problems of design and validation of a comprehensive feature model can be tackled using an integrated process modeling tool, which is generated from a formal metamodel definition.

1.3 Outline
The remainder of this paper is organized as follows: We give an overview over state of the art and state of practice with regards to process tailoring and process model design in Sect. 2. We then introduce the SE Book in Sect. 3, focusing on the parts of the model that are involved in tailoring the process. In Sect. 4 we give an overview over the Process Development Environment, the tool infrastructure that we are using as a basis for the tool presented in this paper. The tool itself is subject of Sect. 5. We conclude the paper in Sect. 6 with a summary and an outlook.

1We did not include a reference as this development process model is property of T-Systems International GmbH and has not been published.
2. RELATED WORK

Rich process models, such as RUP, V-Modell XT [3], or SE Book, are well suited for tool support, as their metamodel [14] can be used to structure the tool’s “data models”. A couple of comprehensive process metamodels (SPEM [15], V-Modell XT [18], ISO/IEC 24744 [5]) have been developed. Yet, there are just a handful of tools to support process engineering: The tool to edit SPEM/EPF is the EF-Composer based on Eclipse. The equivalent tool for the V-Modell XT is the V-Modell XT Editor, which is also used to work with the SE Book. For the ISO 24744 metamodel we are currently not aware of any accepted tool support.

Tailoring and Feature Modeling. In process models based on SPEM, tailoring can be regarded as a “constructive” activity. Customizing the process means to “build” a process model that reflects the organization’s and project’s needs. The tailoring philosophy realized in the SE Book and in the V-Modell XT assumes a ready process model that through the specification of a number of project characteristics gets “pruned” to reflect the needs. Although it seems natural to adapt product line modeling techniques [4] to tailoring of process models, we are not aware of other implementations than the SE Book. In some respect, the tool presented in this paper is a proof of concept for the applicability of Feature-Oriented Domain Analysis (FODA) [6] to process model tailoring.

Discussion. Since, the V-Modell XT and SE Book basically use the same tool for the authoring tasks, the design of a valid tailoring model has to be considered critical. The definition of a tailoring model is an integrated step while working with the process model. Nevertheless, the V-Modell XT Editor does not provide an appropriate support, e.g., to provide support regarding sanity checks [10], or in terms of visual design and assistance. Furthermore, if applying feature modeling for process configuration, process engineers should have an accepted notation available. None of the current process modeling framework (metamodels, tools, etc.) provides that notation. Summarizing the related work, we have a well-known concept, which is yet not applied to the software development process domain. Thus, to implement feature modeling into the considered domain, as well the metamodel as also the corresponding tools have to be created.

3. TAILORING THE SE BOOK

Whenever the SE Book is instantiated for a project, it does exhibit a number of traits depending on the project’s characteristics. It does for example contain rules and regulations about how to deal with sub-contractors if these play a role in the project – otherwise it would not contain those contents. Recommendations for the design of a touch-based user interface may or may not play a role in the project but they certainly do not make sense if the project does not include a user interface at all. Consequently, some of the traits imply or exclude each other.

The SE Book uses a feature model to express the traits of the process model and relations between them. The approach is roughly sketched in Fig. 1: Each feature of the feature tree represents a trait; the instantiated process model may or may not have. The structure of the tree determines the relations between the traits. Each leaf feature is associated with a set of operations to manipulate the process content (currently, the operations are: add a process content element to the resulting process model instance, remove a content element and all its dependencies from the resulting instance). The feature “high safety-requirements” for example leads to the inclusion of a couple of extra activities and artifacts into the resulting instance – process content elements that would otherwise not be part of the process model instance.

The SE Book, including the feature model, associated operations and the process content, is defined in an XML file. Designing the process model basically means to edit the XML file. With over 6 MB the file is so large that it cannot be edited with standard XML editors. The Process Development Environment is a flexible tool that we have built to edit large development process models in general. For the SE Book we built a specialized instance of PDE. The resulting tool is covered in Sect. 5 after a brief description of the PDE platform.

4. THE PDE PLATFORM

The Process Development Environment (PDE) is a platform based on domain-specific languages and has served as a basis for the specialized editor for the SE Book and particularly for its tailoring. It allows the definition of a process metamodel and the generation of a (visual) process model editor. PDE consists of several components, the most important of which are briefly described in the following (for details cf. [8], [17]).

4.1 Technical Foundation

PDE is built using the Microsoft DSL toolkit [2], which provides the basic infrastructure for the definition of DSLs. Metamodels, in terms of DSLs, are developed using a Visual Studio 2010 plugin. Further frameworks are used for certain tasks: The Windows Presentation Foundation (WPF) [13] is the framework used to develop/define the “end-user” application. Runtime plugins are realized using the Managed Extensibility Framework (MEF) [12].

4.2 Platform Architecture

Figure 2 shows the architecture of the PDE. The framework consists of two parts: (1) the PDE Language, which is the extension of the DSL tools, and (2) an PDE Editor.
5. THE SE BOOK IN PDE

The editor tool for the feature model of the SE Book is built on top of the PDE platform and extends the default functionality by adding a custom visual editor for the feature tree and by hooking in to the validation extension points to validate the feature tree.

5.1 Visual Design

As mentioned in Sect. 3, the SE Book is defined in an XML file. If used without extensions, PDE displays the hierarchal structure of the XML file in a tree view with a property grid for the individual elements. This generic user interface works for any kind of process content, independent of its meaning. For the feature-model that is part of the SE Book we have decided to extend default PDE with a visual designer. Figure 3 displays a rough impression of the designer. It uses the accepted notation for feature models and allows the user to manipulate the feature tree with drag and drop.

5.2 Validation

Validation plays an important role in generating consistent product (process) configurations. To ensure a valid feature model, the tool looks for deficiencies in the tree as described in [10]. The paper describes three classes of deficiencies, namely redundancy, anomalies and inconsistencies.

As mentioned in Sect. 3, the SE Book is defined in an XML file. If used without extensions, PDE displays the hierarchal structure of the XML file in a tree view with a property grid for the individual elements. This generic user interface works for any kind of process content, independent of its meaning. For the feature-model that is part of the SE Book we have decided to extend default PDE with a visual designer. Figure 3 displays a rough impression of the designer. It uses the accepted notation for feature models and allows the user to manipulate the feature tree with drag and drop.

By extending the default validation using custom soft constraints, PDE detects all these types of deficiencies by using the feature tree representation generated from the process model DSL. When implementing the validation, we found that some checks seem redundant at first sight, as some deficiencies represent a specialized case of a more general one. In these cases, the tool detects both the general case (e.g., anomaly: “An optional feature is mutual exclusive to a full-mandatory feature”) and the specialized case (e.g., anomaly: “An alternative-child feature is mutual exclusive to a full-mandatory feature”; the alternative-child feature is an optional feature).
The deficiencies in [10] are described in their most simple form, usually for the first “child-parent” level of the feature tree. The SE Book tool also deals with identifying the deficiencies where the source of the problem is located “farther away” in the tree, for example for features situated on different levels in the tree. The reason behind this implementation is given by the previous example, where validation may seem redundant, but in specific configurations only the specialized case can be detected, due to the complexity of the feature model.

If the tool recognizes a deficiency pattern, it outputs a message to the output pane, similar to compiler warnings in an IDE. We have decided to classify the messages as warnings, allowing the user to save a faulty model. We have yet to decide whether to promote some of the deficiencies to errors, which would mean that such deficiencies would have to be resolved before the process model can be saved.

6. CONCLUSIONS AND FUTURE WORK

We have presented a special-purpose tool for editing the SE Book – a proprietary software development process of T-Systems International GmbH. The tool is built using the Process Development Environment (PDE) platform, which in turn is built on top of the Microsoft DSL toolkit.

For a particular area of the development process model, namely the feature-model-based tailoring, we have extended the default PDE functionality to provide a visual editor for the tailoring feature tree and for validation of this tree. The metamodel extension using feature-model-based tailoring is included in the recent release of SE Book. The PDE-based implementation was originally planned for evaluation purposes, and is, still, subject to ongoing research.

6.1 Future Work

We have an implementation for a couple of sanity checks concerning the relation between the feature tree, associated operations and the process content (see Sect. 3 for details). We also have an implementation that translates the feature model into a propositional formula which serves as input into a SAT solver to determine valid configurations [11]. Both implementations are in Java and have yet to be translated to our PDE-based implementation of the process model editor.

While the presented tool offers a comfortable way to edit the tailoring-related parts of the SE Book, other parts of the process model still have to be edited using the default PDE functionality. We are considering to add more visual designers for other parts of the process model, such as the artifact model and the milestone configuration, which means developing a way to reuse visual components from other PDE derivates.

7. REFERENCES


