



Configuration in
Industrial Product Families

ConIPF

<http://www.conipf.org>

IST-2001-34438

Table of Contents

Table of Contents	2
Background	3
Product Family Engineering	4
The ConIPF Methodology	5
Part I: Application Engineering	5
Part II: Domain Engineering	6
Applying ConIPF in Your Organization	7
Experience with the ConIPF methodology	7
Benefits	7
Important considerations	8
Final remarks	9
Contact information	10
Website	10
Robert Bosch GmbH	10
University of Groningen	10
University of Hamburg	10
Thales Nederland	10
Bibliography	11
References	13

Background

Despite all efforts in domain engineering, industrial software product families often face the classic software engineering challenges, such as high costs, long times-to-market, and dependency on product derivation experts, during application engineering. These challenges prevent organizations from exploiting the full benefits of product family engineering.

As a response to these challenges, the ConIPF project (Configuration in Industrial Product Families) was initiated in 2001. ConIPF was funded by the European Union under the IST-programme (contract no. IST-2001-34438). The goal of this project was to define and validate a product derivation methodology that is practicable in industrial application. The project consortium consisted of two industrial partners (i.e. Robert Bosch GmbH and Thales Naval Netherlands) and two academic partners (i.e. the University of Hamburg and the University of Groningen).

One of the results of the ConIPF project is the ConIPF book. This book provides a methodology for product derivation in industrial product families that is designed to address the challenges typical in product family engineering. The basic idea behind the ConIPF methodology is the formalization of derivation knowledge into a configuration model. This configuration model is used to provide automated support during product derivation.

In this whitepaper, we introduce product family engineering in general and present a brief overview of the ConIPF methodology. We furthermore show how your organization can benefit from this methodology and illustrate this with some of our experiences.

Product Family Engineering

The idea behind product family engineering is that the investments required to develop reusable artefacts during domain engineering, are outweighed by the benefits in deriving the individual products during application engineering. A fundamental reason to research and invest in sophisticated technologies for product families is to obtain the maximum benefit out of this upfront investment, in other words, to minimize the proportion of application engineering costs (see Figure 1).

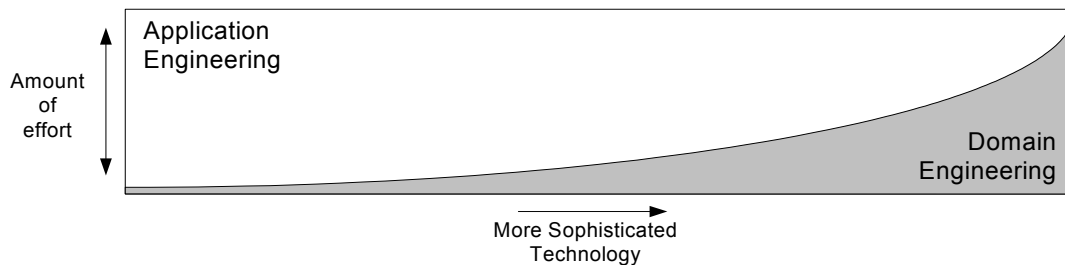


Figure 1: Domain vs. application engineering. The fundamental reason for researching and investing in more sophisticated technology such as product families is decreasing the proportion of application engineering costs.

To achieve this goal, the technology investments can be directed towards domain engineering, or towards application engineering. The ConIPF methodology invests in application engineering and sets a decrease in the costs for deriving individual products as its main goal. The main reason behind this choice is that the industrial organizations with which we work suffer from a lack of methodological guidance for product derivation. This lack of guidance causes a number of challenges that result in the high costs, longer times-to-market and dependency on product derivation experts.

An example of directing investments to domain engineering would be focusing on methodological support for designing and implementing reusable software assets. However, the degree of variability and the need for managing variability during application engineering still remains. Both investments are seen as complementing each other, because the maturity of a product line facilitates the introduction of the ConIPF methodology.

The ConIPF Methodology

The basic idea behind the ConIPF methodology is the formalization of derivation knowledge into a configuration model. This model captures knowledge related to different types of product family entities (e.g. context, features, artefacts, and parameters) as well as relations and restrictions between them. In combination with a configuration tool, some of the artefacts and parameter settings needed to realize a product can be automatically inferred from the selection of the context and the product features.

The ConIPF methodology is presented in two parts. Each part addresses the tasks of the ConIPF methodology that have to be performed in the two stages of product family engineering (see Figure 2), i.e. Application Engineering (I) and Domain Engineering (II).

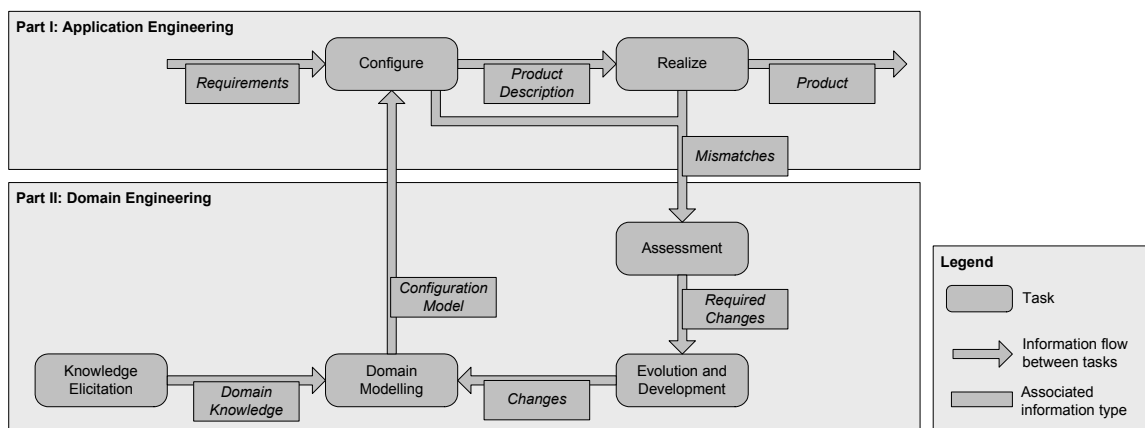


Figure 2: Overview of the main information flow between tasks of the ConIPF methodology.

Part I: Application Engineering

The first part of the methodology discusses how ConIPF uses the configuration model to create individual products. It provides an overview and detailed discussion of the process that is followed to routinely derive the products. This process involves two interrelated main tasks, i.e. *Configure* and *Realize*. The *Configure* task comprises creating different parts of the product description. This product description specifies the context, features, artefacts and parameter values of a product. Under the ConIPF methodology, this task is supported by a configuration tool and the configuration model. Based on the features and context selected by the application engineer, the configuration tool automatically infers the artefacts needed to realize these features in the given context. The *Realize* task realizes the product by actually selecting the artefacts from an asset store and setting the parameters as described in the product description.

For situations that involve choices that cannot be realized according to restrictions in the configuration model, the methodology furthermore describes how Application Engineering interacts with the Domain Engineering. The application engineer can either choose to change some configuration decisions or to collect some of the mismatches that occur during product derivation. The collection of mismatches is passed on to the domain engineer, who is responsible for making the necessary changes to the artefacts and

configuration model. These domain engineering tasks are described in Part II of the methodology.

Part II: Domain Engineering

The second part of the methodology discusses the creation and maintenance of the configuration model. To create the configuration model, the appropriate knowledge has to be selected in the knowledge elicitation task and transformed to a configuration model using the guidelines of the domain-modelling task (Figure 2). For the *Knowledge Elicitation* task, the ConIPF methodology provides a categorization approach for collecting, extracting and loosely structuring derivation knowledge from different sources (documents, experts). It is a step-wise, incremental approach that delivers an informal configuration model. This informal model specifies the basic structure of different concepts involved in product derivation, and is independent of the configuration tool.

During the *Domain Modelling* task, this informal configuration model is transformed to a formal model. The informal model is mapped iteratively to the formal concepts in the ConIPF configuration model. This configuration model is amenable for use by a configuration tool. The domain modelling process also includes testing and evaluation steps to provide confidence with respect to consistency and completeness of the configuration model.

If the model is not consistent, or if mismatches arise during product derivation, the *Assessment* and *Evolution and Development* tasks address identifying the necessary changes to the product family and performing them. The ConIPF methodology accommodates the transfer of these changes to the configuration model by performing change operations in the *Domain Modelling* task. ConIPF does not provide specific rules for the development of actual product family assets however, as this task is highly dependent on the organization. The methodology does provide an *Assessment* process. This process collects the mismatches of the multiple, parallel configuration processes. and provides different steps that divide the assessment challenge into manageable pieces. It identifies the sources of mismatches, and provides support for selecting an optimal solution from a number of solutions with different advantages and disadvantages. After the necessary changes have addressed the mismatches, the application engineering process can continue with the new configuration model.

Applying ConIPF in Your Organization

The ConIPF methodology is largely based on the experiences we gained from its application at two industrial partners in the project, i.e. Robert Bosch GmbH and Thales Naval Netherlands. During experiments at these partners' sites, the methodology was assessed for viability in an industrial setting.

In the following, we first provide an impression of the size and contents of the experiments. Subsequently, we summarize the main benefits of the ConIPF methodology, and present issues that are important to consider when applying the ConIPF methodology.

Experience with the ConIPF methodology

The experiments at Bosch and Thales complemented each other, both in terms of size and focus. The Bosch experiments were performed on the Car Periphery Systems (CPS) product family, which was also used as guiding example throughout the ConIPF book. Although being relatively small (less than hundred features), the configuration model used during the experiments still enables the derivation of hundreds of different product configurations. Compared to the Thales experiments, the Bosch experiments also covered most aspects of the methodology.

The Thales experiments were primarily used to show that the methodology is scalable and adaptable to a specific situation. The Thales experiments consisted of a configuration process that was divided into two layers. One layer concerned the configuration of features and artefacts (several hundreds), while the second layer concerned the configuration of the thousands of parameters. Specific tools for modelling and configuration were used in the experiments at both Bosch and Thales.

Benefits

The ConIPF methodology is designed for existing, mature, and relatively stable (parts of) product families. The experiences with the methodology tell us that once a configuration model is in place for such a product family, the configuration and realization tasks immediately benefit. As the necessary artefacts and parameter values are automatically inferred, the number of explicit decisions is reduced considerably and a large number of common mistakes is avoided.

Automatic inference has a great impact on the time-to-market and cost, as the ConIPF product derivation process involves less iterations than a manual process. However, the ratio between number of products and degree of complexity of the dependencies in the product family are decisive for applying the ConIPF methodology. If the degree of complexity is high, the effort needed for setting up the configuration model pays off even for a small number of products. If the number of products is high, the effort needed for setting up the configuration model pays off even for a low degree of complexity.

As the expert knowledge is captured in the configuration model, it makes their specific and deep expertise available to your organization. The results are that a product can be derived at the features level, and that fewer experts have to be involved in the derivation

process. Their time is thus freed up, and the impact of experts leaving the organization is reduced.

Important considerations

In addition to benefits, the experiments also revealed a number of issues that are important to consider when applying the ConIPF methodology. We summarize these issues in three key points.

- *A good understanding is required:* In the first phase of the experiments, creation of the configuration model was complicated by the fact that the knowledge-based product configuration approach was very new to the software engineers. Adequate explanations and training proved essential to the successful construction of the configuration model. To improve the readability of the ConIPF book for software engineers, we therefore illustrated each aspect with examples from an existing case, and used established notations, such as SPEM (Software Process Engineering Metamodel), in the description of the ConIPF methodology.
- *Tool support is essential:* The experiments showed that mature tool support is essential to the development and maintenance of the configuration model. Tools are available on the market, but sometimes lack the support for evolution, or do not show the rationale behind dependencies and parameter restrictions. This complicates situations where the artefacts and the configuration model have to be adapted. Therefore, the ConIPF book provides a detailed discussion of different tools that can be used for the ConIPF methodology. The application of these different tools ranges from early phases of domain modelling, to the automated inference of artefacts during product derivation.
- *Validation of the configuration model is crucial:* A final important aspect of creating and maintaining the configuration model is associated with building confidence that the configuration model is complete and correct. Building this confidence in domain modelling is similar to testing in software development. It involves expert reviews of the configuration model, as well as creating a test suite. Results from the reviews and tests are used to iterate the domain modelling process. As a result of the experiments, we provide a more detailed and structured description on these activities during domain modelling.

Final remarks

The ConIPF methodology is described in detail in the ConIPF book [Hotz et al. 2005]. Our goal in developing this book was to provide a methodology that addresses all aspects needed to derive products using the ConIPF process. In the book, we provide the tasks and tools to create, maintain, and use a configuration model. What is left for you is to take these tasks and tools, and to apply them to your product family. We believe you will succeed in similar ways as we did in using this methodology.

Contact information

Website

<http://www.conipf.org>

Robert Bosch GmbH

John MacGregor
Robert Bosch GmbH, Department CR/AEA
Eschborner Landstraße 130-132
60489 Frankfurt
Germany
Tel: +49 69 7909 532
Fax: +49 69 9540 295 532

University of Groningen

Drs. Marco Sinnema
University of Groningen, Software Engineering and Architecture Group
Blauwborgje 3
9747 AC Groningen
The Netherlands
Tel: +31 50 363 3939
Fax: +31 50 363 3800

University of Hamburg

Lothar Hotz
HITEC e.V.
% University of Hamburg, Computer Science Department
Vogt-Kölln-Str. 30
22527 Hamburg
Germany
Tel: +49 40 42883 2605
Fax: +49 40 42883 2572

Thales Nederland

Jimmy Troost
Thales Nederland
Haakbergerstraat 49
7554 PA Hengelo
The Netherlands
Tel: +31 74 248 2819
Fax: +31 74 2484046

Bibliography

In addition to the ConIPF book, the consortium produced a total of 30 publications:

- [1] L. Hotz, A. Gunter, *Using knowledge-based configuration for configuring software*, Proceedings of ECAI '02, July 2002.
- [2] J. MacGregor, *Requirements Engineering in Industrial Product Lines*, Proceedings of RE02, September 2002.
- [3] T. Krebs, L. Hotz, A. Gunter, *Knowledge-based configuration for Configuring combined Hardware/Software Systems*, Proceedings of PUK 2002, October 2002.
- [4] L. Hotz, A. Gunter, T. Krebs, *A Knowledge-based Product Derivation Process and some Ideas how to Integrate Product Development*, Proceedings of SVM03, February 2003.
- [5] A. Hein, J. MacGregor, *Managing Variability with Configuration Techniques*, Proceedings of ICSE 2003, May 2003.
- [6] L. Hotz, T. Krebs, *Supporting the Product Derivation Process with a Knowledge-based Approach*, Proceedings of SVM at ICSE 2003, May 2003.
- [7] S. Deelstra, M. Sinnema, J. van Gurp, J. Bosch, *Model Driven Architecture as Approach to Manage Variability in Software Product Families*, Proceedings of the Workshop on Model Driven Architecture: Foundations and Applications (MDAFA 2003), CTIT Technical Report TR-CTIT-03-27, University of Twente, pp. 109-114, June 2003.
- [8] T. Krebs, L. Hotz, *Needed Expressiveness for Representing Features and Customer Requirements*, Proceedings of ECOOP 2003: MVOOPL, July 2003.
- [9] T. Krebs, T. Wagner, W. Runte, *Recognizing User Intentions in Incremental Configuration Processes*, Proceedings of IJCAI'03, August 2003.
- [10] L. Hotz, T. Krebs, *Configuration - State of the Art and New Challenges*, Proceedings of PUK 2003, September 2003.
- [11] T. D. Meijler, S. Schoenmaker, E. de Ruijter, *Modeling in an Architectural Variability Description Language*, Proceedings of PUK 2003, September 2003.
- [12] T. Krebs, L. Hotz, C. Ranze, G. Vehring, *Towards Evolving Configuration Models*, Proceedings of PUK 2003, September 2003.
- [13] K. Wolter, *Orientierung und Navigation im Arbeitsprozess der Produktauswahl von komplexen Produkten*, Proceedings of PUK 2003, September 2003.
- [14] S. Deelstra, M. Sinnema, J. Bosch, *A Product Derivation Framework for Software Product Families*, 5th Workshop on Product Family Engineering (PFE-5), Springer Verlag Lecture Notes on Computer Science Vol. 3014 (LNCS 3014), pp. 473-484, May 2004.
- [15] L. Hotz, T. Krebs, K. Wolter, *Knowledge-based Product Derivation - Research Topics of the ConIPF Project*, Künstliche Intelligenz Heft 4/04, 2004.
- [16] T. Krebs, K. Wolter, L. Hotz, *Mass Customization for Evolving Product Families*, PETO 2004, June 2004.
- [17] L. Hotz, T. Krebs, K. Wolter, *Using a Structure-based Configuration Tool for Product Derivation*, Proceedings of ASE 2004, September 2004.

- [18] L. Hotz, T. Krebs, K. Wolter, *Dependency Analysis and its Use for Evolution Tasks*, Proceedings of PUK 2004, September 2004.
- [19] M. Sinnema, S. Deelstra, J. Nijhuis, J. Bosch, COVAMOF: A Framework for Modeling Variability in Software Product Families, Proceedings of the Third Software Product Line Conference (SPLC 2004), Springer Verlag Lecture Notes on Computer Science Vol. 3154 (LNCS 3154), pp. 197-213, August 2004.
- [20] S. Deelstra, M. Sinnema, J. Nijhuis, J. Bosch, *Experiences in Software Product Families: Problems and Issues during Product Derivation*, Proceedings of the Third Software Product Line Conference (SPLC 2004), Springer Verlag Lecture Notes on Computer Science Vol. 3154 (LNCS 3154), pp. 165-182, August 2004.
- [21] T. Krebs, L. Hotz, K. Wolter, *Pre-Packaged Variability for Product Derivation in Product Lines*, Proceedings of ECAI 2004: WS Configuration, August 2004.
- [22] K. Wolter, T. Krebs, L. Hotz, T. D. Meijler, *Knowledge-based Product Derivation Process*, AIAI 2004, August 2004.
- [23] M. Sinnema, O. de Graaf, J. Bosch, *Tool support for COVAMOF*, Proceedings of the Workshop on Software Variability Management for Product Derivation - Towards Tool Support, August 2004.
- [24] L. Hotz, T. Krebs, K. Wolter, *Combining Software Product Lines and Structure-based Configuration – Methods and Experiences*, Proceedings of the Workshop on Software Variability Management for Product Derivation - Towards Tool Support, August 2004.
- [25] S. Deelstra, M. Sinnema, J. Nijhuis, J. Bosch, COSVAM: A Technique for Assessing Software Variability in Software Product Families, Proceedings of the 20th IEEE International Conference on Software Maintenance (ICSM 2004), pp. 458-462, September 2004.
- [26] L. Hotz, *Methods for Knowledge-based Product Derivation*, poster presentation at ICSR 2004, November 2004.
- [27] M. Sinnema, S. Deelstra, J. Nijhuis, J. Bosch, *Managing Variability in Software Product Families*, Proceedings of the 2nd Groningen Workshop on Software Variability Management, December 2004.
- [28] S. Deelstra, J. Nijhuis, J. Bosch, M. Sinnema, *The COVAMOF Software Variability Assessment Method (COSVAM)*, Proceedings of the 2nd Groningen Workshop on Software Variability Management, December 2004.
- [29] S. Deelstra, M. Sinnema, J. Bosch, *Product Derivation in Software Product Families; A Case Study*, Journal of Systems and Software, Vol 74/2 pp. 173-194, January 2005.
- [30] T. Krebs, K. Wolter, L. Hotz, *Model-based Configuration Support for Product Derivation in Software Product Families*, IMCM 2005, June 2005.

References

- [Hotz et. al 2005]. HOTZ, L. ET AL.: *Configuration in Industrial Product Families - The ConIPF Methodology*, Berlin: AKA-Verlag, to appear, 2005.