

Support for configuration of physical products and services

Juha Tiihonen



Support for configuration of physical products and services

Juha Tiihonen

A doctoral dissertation completed for the degree of Doctor of Science (Technology) to be defended, with the permission of the Aalto University School of Science, at a public examination held at the lecture hall TU1 of the school on 31 October 2014 at 12.

Aalto University
School of Science
Department of Computer Science and Engineering
Product Data Management Group

Supervising professor

Casper Lassenius

Thesis advisors

Tomi Männistö

Reijo Sulonen

Preliminary examiners

Cipriano Forza, Università degli Studi di Padova, Italy

Asko Riitahuhta, Tampere University of Technology, Finland

Opponent

Gerhard Friedrich, Alpen-Adria Universität Klagenfurt, Austria

Aalto University publication series

DOCTORAL DISSERTATIONS 153/2014

© Juha Tiihonen

ISBN 978-952-60-5894-8

ISBN 978-952-60-5895-5 (pdf)

ISSN-L 1799-4934

ISSN 1799-4934 (printed)

ISSN 1799-4942 (pdf)

<http://urn.fi/URN:ISBN:978-952-60-5895-5>

Unigrafia Oy

Helsinki 2014

Finland



Author

Juha Tiihonen

Name of the doctoral dissertation

Support for configuration of physical products and services

Publisher School of Science**Unit** Department of Computer Science and Engineering**Series** Aalto University publication series DOCTORAL DISSERTATIONS 153/2014**Field of research** SCI020Z Computer Science and Engineering**Manuscript submitted** 15 April 2014**Date of the defence** 31 October 2014**Permission to publish granted (date)** 30 September 2014**Language** English **Monograph** **Article dissertation (summary + original articles)****Abstract**

The ideal of mass customization is to satisfy individual customer requirements efficiently. This can be realized with configurable products that can be adapted, within the bounds of pre-designed offered variation, to individual requirements. Configurators are information systems that support efficient and errorless specification of individualized products or services.

Applying the Design Science research approach, a number of artifacts that support the sales configuration of physical products and services were developed with the aim of advancing the state of the art of practically applicable configurators.

We present a conceptualization for configuration knowledge that unifies previous connection-based, resource-based, structure-based, and function-based approaches. The conceptualization is object oriented and treats the main concepts uniformly with respect to several criteria, including the availability of taxonomic hierarchies with refinement, abstraction, and applicability of attributes. A detailed conceptualization for representing the variable compositional structure of components and functions is included.

The main artifact of this work is a novel configurator instantiation called WeCoTin. It consists of a graphical modeling environment Modeling Tool and a Web-based Configuration Tool that supports the configuration task. WeCoTin is based on a well-founded modeling conceptualization and a corresponding high-level object-oriented modeling language with clear formal semantics, provided by mapping the modeling language to weight constraint rules—a form of logic programs. The Modeling Tool enables efficient graphical modeling and includes features that support long-term management. The Configuration Tool provides Web-based sales configuration functionality. It applies an inference engine that is based on weight constraint rules to provide consistent and complete inference. Evaluation includes the characterization of 26 sales configuration models and run-time performance analysis. Enabled by the novel principles of WeCoTin, an information systems design theory for sales configurators is proposed.

Recommendation technologies can support users during choice navigation. We present scenarios in which support can be offered and analyze the applicability of recommendation technologies. We propose extensions to the existing case-based feature value recommendation technologies by integrating importance weights and similarity metrics. A basic evaluation of the utility of the case-based collaborative approach was provided through an empirical study.

Due to the importance of services, mass customization of services by configuration is crucial. We discuss the offered variation of configurable services in three industries and the applicability of configurators designed for physical products in the context of service configuration.

Keywords configurator, recommender, recommendation, services, service configuration, configuration modeling, Design Science, information systems design theory

ISBN (printed) 978-952-60-5894-8**ISBN (pdf)** 978-952-60-5895-5**ISSN-L** 1799-4934**ISSN (printed)** 1799-4934**ISSN (pdf)** 1799-4942**Location of publisher** Helsinki**Location of printing** Helsinki**Year** 2014**Pages** 112**urn** <http://urn.fi/URN:ISBN:978-952-60-5895-5>

Tekijä

Juha Tiihonen

Väitöskirjan nimi

Fyysisten tuotteiden ja palveluiden konfiguroinnin tietotekninen tuki

Julkaisija Perustieteiden korkeakoulu**Yksikkö** Tietotekniikan laitos**Sarja** Aalto University publication series DOCTORAL DISSERTATIONS 153/2014**Tutkimusala** SCI020Z Tietotekniikka**Käsitteilyajon pvm** 15.04.2014**Väitöspäivä** 31.10.2014**Julkaisuluvan myöntämispäivä** 30.09.2014**Kieli** Englanti **Monografia** **Yhdistelmäväitöskirja (yhteenvedo-osa + erillisartikkelit)****Tiivistelmä**

Massaräätälöinnin tavoitteena on täyttää yksilölliset asiakasvaatimukset tehokkaasti. Tämä voidaan saavuttaa konfiguroitavilla tuotteilla, joita voidaan muunnella yksilöllisiin tarpeisiin etukäteen tehdyn suunnittelun sallimissa rajoissa. Konfiguraattorit ovat tietojärjestelmiä, jotka tukevat tehokasta ja virheetöntä yksilöllisten tuotteiden tai palvelujen määrittelyä.

Design Science -lähestymistapaa soveltaen työssä kehitettiin artefakteja, jotka tukevat fyysisten tuotteiden sekä palvelujen myyntikonfigurointia. Tarkoituksena oli mahdollistaa entistä parempia ja käytännössäkin sovellettavissa olevia konfiguraattoreita.

Työssä esitetään konfigurointitietämyksen käsitelmä, joka yhdistää aiemmat mallitustavat, jotka perustuvat kytkentöihin, resursseihin, rakenteeseen ja toimintoihin. Oliopohjainen käsitelmä soveltaa pääkäsitteille yhtenäisesti luokittelua, abstraktiota ja attribuutteja. Käsitelmässä on yksityiskohtainen muunneltavan kompositionaalisen rakenteen mallitus.

Työn tärkein artefakti on uudenlainen konfiguraattoritoteutus WeCoTin. Se koostuu graafisesta mallitusympäristöstä (Modeling Tool) ja verkon yli konfigurointia tukevasta työkalusta (Configuration Tool). Järjestelmän hyvin määritelty mallituskäsitteistö on osajoukko konfigurointitietämyksen käsitelmästä. Mallituskielellä on selkeä formaali semantiikka, joka perustuu käännökseen logiikkapohjaiseen tiedonesittämiskieleen Weight Constraint Rule Language (WCRL). Modeling Tool tukee tehokasta graafista konfigurointimallitusta ja sisältää pitkäaikaishallintaa tukevia ominaisuuksia. Configuration Tool tukee myyntikonfigurointia verkon kautta. Sen WCRL-pohjainen päättelykone tarjoaa täydellisen ja ristiriidattoman päättelyn. WeCoTinta on arvioitu 26 konfigurointimallin luomisella ja karakterisoinilla sekä testaamalla ajonaikaista suorituskykyä. WeCoTin-konstruktion uusien toteutusperiaatteiden pohjalta esitetään myyntikonfiguraattorien suunnitteluteoria (information systems design theory).

Suositteluteknologiat voivat tukea käyttäjiä kun he tekevät valintoja konfiguroitaessa. Työssä tunnistetaan tilanteita, joissa suosittelusta voi olla hyötyä ja analysoidaan tärkeimpien suositteluteknologioiden soveltuvuutta. Eräisiin aiempiin suositteluteknikoihin esitetään laajennuksia, jotka perustuvat valintojen tärkeyteen käyttäjälle (painoarvo) ja samankaltaisuutta esittävien metriikoiden soveltamiseen (similarity metrics). Suositteluteknikoiden tarjoama hyöty käyttäjille osoitetaan alustavasti empiirisellä kokeella.

Palvelujen tärkeys korostaa myös palvelujen massaräätälöinnin ja konfiguroinnin tärkeyttä. Työssä käsitellään palvelujen muunneltavuutta kolmella toimialalla sekä fyysisille tuotteille soveltuvien konfiguraattorien soveltuvuutta palvelujen konfigurointiin.

Avainsanat konfiguraattori, suositusjärjestelmä, suositelu, palvelut, palvelujen konfigurointi, konfiguraatiomallitus, Design Science, tietojärjestelmien suunnitteluteoria

ISBN (painettu) 978-952-60-5894-8**ISBN (pdf)** 978-952-60-5895-5**ISSN-L** 1799-4934**ISSN (painettu)** 1799-4934**ISSN (pdf)** 1799-4942**Julkaisupaikka** Helsinki**Painopaikka** Helsinki**Vuosi** 2014**Sivumäärä** 112**urn** <http://urn.fi/URN:ISBN:978-952-60-5895-5>

Preface

This thesis was prepared in the Product Data Management Group (PDMG) of Aalto University (formerly Helsinki University of Technology). The PDMG's research interest is "how information technology can be used for managing complex products with many variants and long life cycles." Configuration has been one of the main streams of PDMG research since the early 1990s.

This work covers a long time span. The author began configuration-related research in late 1992. The initial phases concentrated on understanding configuration problems and management of product data. Conceptual modeling of the central phenomena of configuration and product evolution was researched. An idea occurred: Why not implement a configurator that would capture PDMG's understanding about practical configuration problems, provide adequate support for end users, perform high-level modeling, and make use of a state-of-the-art inference engine developed in a neighboring research group? A research project was established to develop a concrete system called WeCoTin (Web Configuration Technology). Research around WeCoTin forms the main trunk of this thesis.

Later, the importance of services was recognized, creating a need to understand the relation of services to mass customization and configuration. The newest branch of research in this thesis studies recommendation technologies in the context of product configurators. Industrial research partners described a need for *consultative sales*. Here, the seller actively constructs and recommends solutions to genuinely fit the needs and goals of the customer.

I wish to express my gratitude to all previous and current members of PDMG. Professor Tomi Männistö provided most useful support as an instructor during the last phases of this work. His ideas and inspiration have affected many aspects of this work. Professor Reijo Sulonen, as the supervisor during most of this work, earns my gratitude for his guidance and patience and for identifying fascinating topics of research. Professor Casper Lassenius provided valuable guidance and motivation as the supervisor as this thesis was finalized.

The comments of the pre-examiners Cipriano Forza and Asko Riitahuhta were invaluable.

Timo Soininen was an inspiring colleague for intensive discussions and debates and the leader of a number of related research projects. Asko Martio provided invaluable practical understanding about the challenges of industrial companies related to management of configurable offerings. He and Reijo Sulonen acquired a number of cases needed for the evaluation of WeCoTin.

Alexander Felfernig provided a jump start to research on the recommendation of configurable offerings, and his cooperation has been fruitful ever since.

Andreas Anderson was the wizard who made WeCoTin a reality with his extraordinary skills of systems development. Nothing was ever difficult. Hannu Peltonen has a talent for thinking clearly and developing systems based on his clear thought. This was extremely useful for specifying conceptualizations and the corresponding core of a configuration modeling system that later evolved into WeCoTin. Mikko Heiskala was a key member of the WeCoTin development team, and his expertise in service modeling has been invaluable. Kaija-Stiina Paloheimo challenged many basic assumptions of thinking about services. Matti Sievänen provided invaluable understanding about cost management and pricing of configurable offerings. I guess it was Matti who indoctrinated me with the idea of cycling during our frequent meetings. This significantly changed my life.

I am grateful to individuals who were part of the WeCoTin development team or modeled products with it: Sami Asikainen, Miguel Luis Ferreira, Jan Elfström, Tero Kojo, Juha-Miikka Nurmilaakso, Mikko Pasanen, Kati Sarinko, and Johanna Voutilainen. I thank the student group Dotcomrades for designing and implementing WeCoTin's graphical constraint editor (Sakari Ailus, Pete Hakkarainen, Mika Koskimäki, Jukka Parviainen, Thach Pham, Panu Ranta, and Juho Tikkala). I thank the student group CCCP for designing and implementing a tool for comparing configurable products (Andreas Anderson, Lasse Anderson, Juha Havu, Mikko Heiskala, Virpi Huhtinen, Matias Karvinen, and Anni Toikka).

Monika Mandl implemented the RECOMOBILE environment with considerable effort. Juha Vepsäläinen checked the statistical conclusions of this thesis.

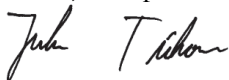
Johanna "Jonna" Lehtola shed light on darkness at some phases of this work. Juha Laine, Casper Lassenius, Marjo Kauppinen, and Marko Nieminen all encouraged me to just say "it's ready." I needed that. All my great colleagues and support team members, thank you for making SoberIT such an inspiring and pleasant work environment. I want to thank all the other people who contributed in one way or another.

Research funding from the Technology Development Centre of Finland (TEKES) is gratefully acknowledged. We thank Gardner Denver Finland, KONE, Patria, Tapiola Group, Tamrock, and a number of undisclosed companies for sharing product information. We also thank Aalto Service Factory for funding.

Many of you have become good friends, which may be even more important than support of substance.

Last but not least, I want to thank Marja for her loving support and patience, Iiro and Olli, and my parents, relatives, and friends for being there.

Helsinki, 22 September 2014



Juha Tiihonen

Contents

Preface	1
List of Abbreviations	5
List of Publications	8
Author's Contribution and Publication Information.....	9
1. Introduction	11
1.1 Background	11
1.2 Research questions	14
1.3 Methodology and overview of contributions	14
1.4 Structure of the thesis.....	19
2. Previous work	23
2.1 Overview of literature.....	23
2.2 Basic terminology	23
2.3 Configuration knowledge modeling.....	24
2.4 Configurators	28
2.5 Recommendation of configurable offerings	31
2.6 Mass customization and configuration of services.....	34
3. Conceptualization for configuration knowledge.....	37
3.1 Types, individuals, and classification	37
3.2 Attributes	38
3.3 Component types and compositional structure—parts.....	39
3.4 Topology—ports	39
3.5 Resources	40
3.6 Functions	40
3.7 Constraints.....	40
3.8 Example	41
4. WeCoTin Configurator	43
4.1 Requirements.....	43
4.2 WeCoTin overview.....	45
4.3 WeCoTin Configuration Tool	45

4.4	Modeling Tool and PCML.....	45
4.5	User interface modeling and generation.....	48
4.6	Determining price and delivery time	49
4.7	Inference with weight constraint rules and Smodels	49
5.	Recommendation of configurable offerings	51
5.1	Scenarios for recommending configurable offerings	51
5.2	Selection of recommendation techniques.....	52
5.3	Proposed extensions to recommendation algorithms.....	53
6.	Sales configuration of services	59
6.1	Offered variation in configurable service products	59
6.2	Sources of variation in relationship-based services	61
6.3	Services and configurators for physical products.....	62
7.	Evaluation	65
7.1	Artifact evaluation	66
7.2	Sales Configurator Information Systems Design Theory.....	74
7.3	This work and the guidelines of Design Science.....	78
8.	Discussion	81
8.1	Related work	81
8.2	Threats of validity	85
8.3	Answers to the research questions and contributions	87
8.4	Future research.....	92
9.	Conclusions	93
	References	96

List of Abbreviations

ASP	Answer set programming (ASP) makes it possible to express a problem as a theory consisting of logic program rules with clear declarative semantics, and the stable models of the theory correspond to the solutions (answer sets) of the problem. The Smodels system applied in this work follows the ASP paradigm.
B2C	Business-to-consumer.
BCRL	Basic constraint rule language. The output of grounding a logic program expressed in WCRL is a corresponding logic program expressed in BCRL. It contains no variables and can be directly used by an inference procedure such as smodels.
CSP	Constraint satisfaction problem. A constraint satisfaction problem is a tuple (V, D, C) . Here, V is a set of finite domain variables, $V = \{v_0, v_1, \dots, v_n\}$. Each variable has a (usually finite) domain that specifies the possible values of the variable, and the set of domains is D , $D = \{dom_0, dom_1, \dots, dom_n\}$. C is a set of constraints specifying restrictions on the allowed combinations of variable value assignments. A solution to a constraint satisfaction problem is a set of assignments to each variable $\{v_0 = x_0, v_1 = x_1, \dots, v_n = x_n\}$ such that each $x_i \in dom_i$ and the assignments are consistent with the set of constraints C (Mackworth & Freuder, 1985). Approaches based on CSP formulations are common for problem solving of configurators.
DCSP	Dynamic constraint satisfaction problem (Mittal & Falkenhainer, 1990; Sojininen & Gelle, 1999). A DCSP allows dynamic activation and deactivation of variables and constraints of a CSP.
GCSP	Generative CSP (Fleischanderl, Friedrich, Haselböck, Schreiner, & Stumptner, 1998; Stumptner, Friedrich, & Haselböck, 1998). A GCSP allows dynamic generation of new variables and instantiation of related constraints in a CSP.
ID	(Unique) identifier.
IHIP	Services are often attributed with the characteristics of intangibility, heterogeneity, inseparability of production and consumption, and perishability, collectively known as IHIP (Grönroos, 2007; Zeithaml, Parasuraman, & Berry, 1985).
IS	Information systems (research discipline): "Its mission is to advance knowledge about the effective and efficient utilization of information technology by individuals, groups, organizations, soci-

ety, and nations for the improvement of economic and social welfare” (ISR, 2013).

ISDT	Information systems design theory is the primary output of Design Science research that “shows the principles inherent in the design of an IS artifact that accomplishes some end, based on knowledge of both IT and human behavior” (Gregor & Jones, 2007, p.322).
IT	Information technology.
lparse	Component lparse of the Smodels system is a front end that compiles a WCRL program with variables into programs in the basic constraint rule language (BCRL) that contains no variables.
MAUT	Multi-attribute utility theory, a decision support method (Dyer, 2005; Von Winterfeldt & Edwards, 1986).
PC	Personal Computer.
PCML	Product Configuration Modeling Language is used in WeCoTin configurator as the language for representing configuration models. PCML is object oriented and declarative and has formal implementation-independent semantics.
PDMG	Product Data Management Group (PDMG) is a research group of Aalto University (formerly Helsinki University of Technology). The main research interest of PDMG is “How information technology can be used for managing complex products with many variants and long life cycles.” The author of this thesis is a member of PDMG.
RECOMOBILE	A prototype configurator system with recommendation functionality (IV, Felfernig, Mandl, Tiihonen, & Schubert, 2010).
RQ	Research question. Often in the form RQn: for example, RQ1 is research question 1.
SCISDT	Sales configurator information systems design theory is a partial ISDT for sales configurators proposed in this work. It is based on the design of WeCoTin.
SCML	Service Configuration Modeling Language. SCML enables the expression of configuration models of services with a PCML-like syntax and improved conceptual match (Anderson, 2005).
Smodels	The Smodels system (Simons, Niemelä, & Soininen, 2002) provides an efficient inference engine for WCRL. It consists of components lparse and smodels.
smodels	Component smodels of the Smodels system provides the main functionality of Smodels—to compute a desired number of stable models for a BCRL program. Requirements are specified through <i>compute statements</i> to constrain the stable models to be computed.
UML	Unified Modeling Language (e.g. Rumbaugh, Jacobson, & Booch, 1999).

- WCRL Weight constraint rule language allows expression of a problem as a theory consisting of logic program rules with clear, declarative semantics. WCRL is equipped with weight constraints for representing weighted choices with lower and upper bounds and with conditional literals restricted by domain predicates to encode sets of atoms over which the choices are made.
- WeCoTin A Web-based configurator prototype (abbreviation for Web Configuration Technology). WeCoTin consists of two main components: the graphical modeling environment Modelling Tool and the Web-based Configuration Tool that supports the configuration task.
- XML Extensible Markup Language (W3C, 2008) defines a relatively simple and general way to encode documents in a format that is readable to both humans and machines and widely applied (Wikipedia, 2014). For a good overview, see (Wikipedia, 2014).

List of Publications

This doctoral dissertation consists of a summary and of the following publications which are referred to in the text by their Roman numerals:

- I** Soininen, T., Tiihonen, J., Männistö, T. & Sulonen, R. (1998). Towards a general ontology of configuration. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing (AI EDAM)*, 12(4), 357–72.
- II** Tiihonen J., Heiskala M., Anderson A., and Soininen T. (2013). WeCoTin—a practical logic-based sales configurator. *AI Communications*, 26(1), 99–131. doi 10.3233/AIC-2012-0547.
- III** Tiihonen J., Felfernig A. (2010). Towards recommending configurable offerings. *International Journal of Mass Customisation*, 3(4), 389–406.
- IV** Felfernig, A. Mandl, M., Tiihonen, J. Schubert, M., and Leitner G. (2010). Personalized user interfaces for product configuration. 15th International Conference on Intelligent User Interfaces (IUI '10), Hong Kong, China, 317–20. doi 10.1145/1719970.1720020.
- V** Tiihonen J., Heiskala M., Paloheimo K.-S., Anderson A. (2006). Configuration of contract based services. 17th European Conference on Artificial Intelligence (ECAI 2006), configuration workshop, Riva del Garda, Italy, 25–30.

Author's Contribution and Publication Information

Publication I: Towards a general ontology of configuration

The author of this thesis was the second author. He originated the idea of writing the paper, contributed to the development of the concepts (especially the connection-oriented and resource-oriented concepts), and was the main developer of the example. In addition, he contributed requirements, experiences, and ideas on compositional structure and attributes based on his experiences from an M.Sc. thesis project (Tiihonen, 1994). Soinen as the first author was the principal developer of the configuration conceptualization (called *ontology* in the publication), especially the foundation on Ontolingua and formalization, the distinction between types and individuals, and the conceptualizations of attributes, compositional structure, functions, contexts, and constraints. The other authors reviewed and polished the conceptualization. This publication was included in Soinen's doctoral thesis (2000).

Publication II: WeCoTin—a practical logic-based sales configurator

This article summarizes research around the WeCoTin configurator. It was entirely written by the author of this thesis. Heiskala reviewed and helped polish the text. Heiskala, Anderson, and Soinen were included as authors on account of their previous contributions to WeCoTin and related publications. Further details on WeCoTin contributors are appended at the end of the publication.

Publication III: Towards recommending configurable offerings

The author of this thesis was the principal author. Felfernig identified previous work on recommendation systems and distance metrics, suggested including the nearest-neighbor algorithm, suggested numerous improvements, and provided general guidance.

Publication IV: Personalized user interfaces for product configuration

The author of this thesis was the third author, was responsible for the recommendation algorithms applied in RECOMOBILE (implementation and integration of an idea from Publication III), and provided the example configuration model of mobile subscriptions and phone selection. The author also actively participated in formulating the research hypotheses with Felfernig and Mandl. The

experiment was designed and the results were analyzed and documented mainly by Felfernig and Mandl. The peer-review process based on the full paper had four reviewers. The acceptance rate was 30% (62 of 206 submissions).

Publication V: Configuration of contract-based services

The author of this thesis was the principal author, performed the main body of analysis, and documented the results. Other authors were equally involved in the underlying case studies. Paloheimo and Heiskala contributed most of the method and previous work. The peer review process was based on full paper submission. There were three reviewers, but the acceptance rate was not published.

Subcontracting: Language check

The author considered all suggested changes of the language check. No other subcontracting took place.

1. Introduction

1.1 Background

Mass customization strategy and configurable products

Mass customization aims to provide products or services that closely match the individual needs of customers while retaining mass-production-like efficiency (Pine, 1993). To be successful in business based on mass customization strategy, mass customizers require three fundamental capabilities (Salvador, de Holan, & Piller, 2009): First, *solution space development* is required to identify the product properties¹ for which customer needs diverge. Second, a *robust process* enables reuse or recombining of existing organizational and value-chain resources to fulfill a stream of differentiated customer needs. Finally, *choice navigation capability* is required to support customers in identifying their own solutions while minimizing complexity and the burden of choice.

Figure 1 illustrates several main concepts² and their relationships. Business based on configurable products is one way of achieving mass customization. A *configurable product* is the result of solution space development; a design of a family of products that is provided in advance and can be adapted to meet diverging customer requirements within the scope of designed possibilities of adaptation. A *configuration task* produces a specification of a *product individual* that meets *customer requirements* and conforms to the rules of the configurable product. A specification of a *product individual* based on a configurable product is called a *configuration*. A configuration specifies one of the possible product *variants*. Variants are based on a family of products but differ from each other in terms of some properties. If services are included in a configuration, a service process (consisting of activities and resources) may also be specified by the configuration. *Offered variation (variation for short)* denotes the selection space available to customers – what variable properties are selectable.

The product family description containing all the information on the possibilities of adapting the configurable product to customer needs is called a *configuration model*. A configuration model specifies the entities that can appear in a configuration, their properties, and the rules on how the entities and their properties can be combined. The configuration model is expressed with mod-

¹ *Property* is defined in Section 2.2.

² *Concept* is defined in Section 2.2.

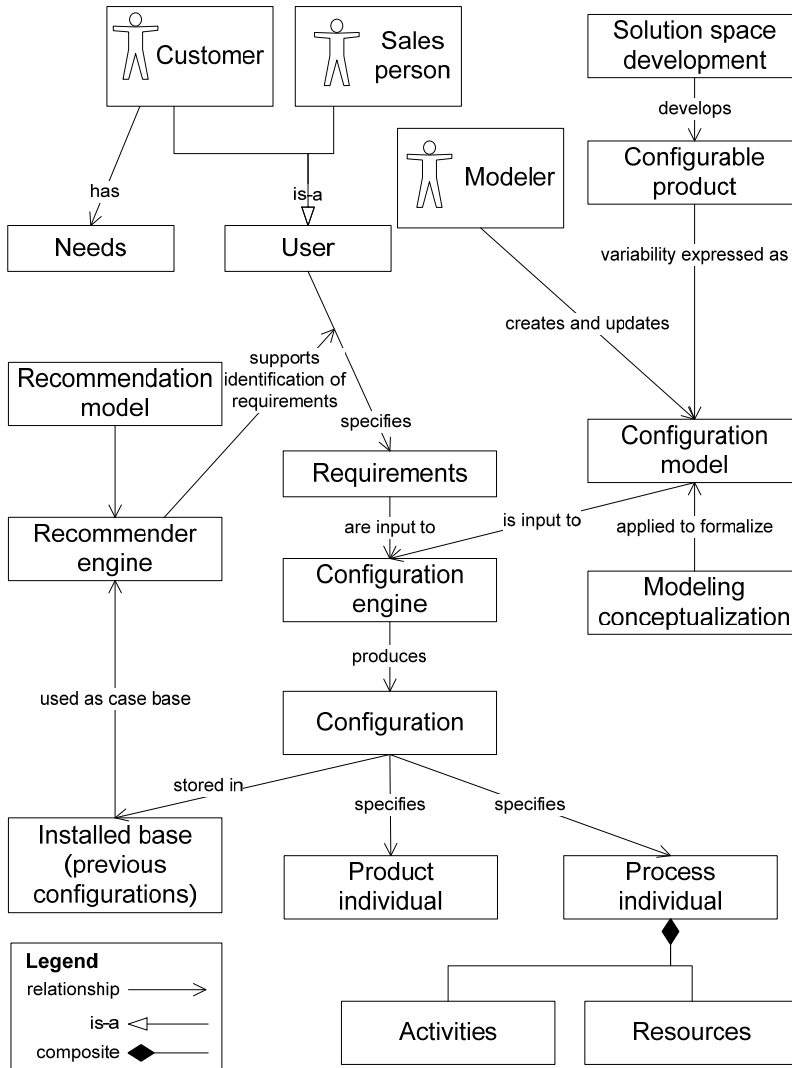


Figure 1. Core concepts and their major relations.

eling concepts supported by the applied *conceptualization*. Adopting Gruber’s definition (1993), a *conceptualization* is an abstract, simplified view of the world that we wish to represent for some purpose that consists of the objects, concepts, and other entities that are presumed to exist in some area of interest and the relationships between them. For example, some configuration conceptualizations support *component types*,³ *attributes*, and *constraints*.

The potential benefits and challenges of mass customization and configurable products are significant (see Heiskala, Paloheimo, & Tiihonen, 2007).

³ See Sections 2.2 and 3.3 (component type) for definitions.

Many of the challenges can be addressed with information technology (IT) support because of the well-defined nature of the configuration task.⁴

Configurators and recommendation support

A class of systems, *configurators*, makes it possible to represent the offered variation of configurable products by creation and management of configuration models, as well as to support users in performing the configuration task. Configurators are often critical to companies that base their business on mass customization, because they provide the essential choice navigation capability.

One or more *modelers* create and maintain configuration models. The typical *user* of a configurator is directly a *customer* or a *salesperson* who aims to sell an individualized product or service to the customer.

Informal *customer needs* are formally represented as *requirements*. Based on requirements and the configuration model, the configuration engine produces a configuration that specifies the product individual that meets the customer requirements. Thus, a configurator can form the basis for the choice navigation capability of a company.

Often customers neither know their exact requirements in advance nor know which of the available alternatives fit their preferences. Therefore, assistance may be desirable. Such assistance can be provided by a competent salesperson. An emerging alternative is to integrate recommendation functionality into a configurator's user interface, e.g., to suggest suitable alternatives. Previous configurations and a *recommendation model* may be used to identify recommendable alternatives or requirements or to resolve conflicting requirements. A recommendation model may include explicit knowledge on recommendable (and not recommendable) aspects as well as policies or methods of deriving such information.

Services are an important part of the global economy. Some services, such as maintenance contracts, telecommunications services, portfolios of financial investments, and insurance policies, can be configured. There is a need to understand the relation between services and mass customization as well as the impacts of services on configurators.

Numerous configurators have been developed both as research prototypes and as commercial software. The landmark R1/XCON was deployed at Digital Equipment Corporation in the early 1980s (McDermott, 1982), and experiences, benefits, and challenges of using it have been widely documented; see, e.g., Sviokla (1990) or McDermott (1993).

Major research efforts have been undertaken to provide configurators applicable to solving general configuration tasks instead of a specific domain, e.g., Frayman and Mittal (1987), Cunis, Günter, Syska, Peters & Bode (1989), and

⁴ Advance solution space design implies that one does not design new component types or novel ways of combining component individuals to satisfy arbitrary customer requirements. Sometimes a significant part of the product is configured, but creative or innovative design may be needed to meet requirements outside the designed solution space; this mode of operation with 'partially configurable products' is important to many companies (Tiihonen, 1999; Tiihonen, Soininen, Männistö, & Sulonen, 1998). Partially configurable products are beyond the scope of this thesis.

Stumptner, Haselböck & Friedrich (1994). A large number of commercial general-purpose knowledge-based configurators aka *configuration frameworks* exists (e.g. Sabin & Weigel, 1998); Anderson (2005) identified 30 vendors based on their Web pages. In addition, it is customary that prominent enterprise resource planning systems include a configurator module as documented by Haag (1998) and Damiani, Brand, Sawtelle & Shanzer (2001).

Configurators are deployed relatively widely. For example, 970 Web-based configurators were listed in the International Configurator Database (cyLEDGE, 2013).

1.2 Research questions

The research theme that unifies research questions of this work is: *How to effectively support configuration of physical products and services?* To limit the scope of the thesis and adhere to the research opportunities identified in Section 2, the following more focused research questions are addressed:

- RQ1: What are the concepts central to configuration knowledge?
- RQ2: How to construct a practical and computationally well-founded sales configurator?
- RQ3: Can users be effectively supported in finding suitable products and services with personalized recommendations?
- RQ4: How does service configuration differ from the configuration of physical products?

1.3 Methodology and overview of contributions

Sciences of the artificial (Simon, 1996) is concerned with artificial things. Artificial things a.k.a. *artifacts* (Simon, 1996) are elements synthesized or constructed by humans.⁵ They can be characterized in terms of functions,⁶ goals, and adaptation to “outer environment.” Furthermore, artifacts can be discussed in terms of imperatives, especially during design time, which reflects the idea that requirements can be specified and taken as design targets.

Simon (1996) discusses the fundamentals of the science of the artificial with a wide scope. He identifies a non-exhaustive list of necessary topics in theory of design. He covers the fundamental properties of natural and artificial worlds, economic and bounded rationality, the psychology of thinking and decision-making, and problem-solving and role of representations in problem solving. Further topics include taking the future and society into account in the design, the role of the designer, and complexity and management of complexity through hierarchic systems. However, Simon does not present an actual theory or methodology of design.

⁵ Other meanings of ‘artifact’ are outside the scope of this thesis.

⁶ Concept ‘function’ is discussed in Section 2.3.2.

Configurable products, their modeling and supporting tools are in the domain of the sciences of the artificial. The primary interest of this thesis is on information technology (IT) artifacts for managing configurable products and services. Simon's (1996) ideas underlie the Design Science approach that creates and evaluates IT artifacts intended to solve identified organizational problems (Hevner, March, Park, & Ram, 2004). First, Design Science will be briefly described (Section 1.3.1), and then this work will be characterized as an instance of Design Science research, and its contributions briefly outlined (Section 1.3.2). In some aspects of this work, Design Science was augmented with the case study research method (Yin, 2009), summarized in Section 1.3.3.

1.3.1 Design Science

Hevner et al. (2004) characterize the Design Science approach as follows (Figure 2). The *environment* defines the problem space in which the phenomena of interest reside. In Information systems (IS) research, the environment consists of *people*, *organizations*, and *technology*. People in an organization perceive, assess, and evaluate *business needs* in the environmental context of their organization. The business needs perceived by the researcher stem from this context. Research *relevance* is assured by framing research to address business needs.

Design Science research is conducted through *building* and *evaluation of artifacts* designed to meet the identified business need, the ultimate goal being utility. The artifacts can be *constructs* (vocabulary and symbols), *models* (abstractions and representations), *methods* (algorithms and practices), or *instantiations* (implemented or prototype systems). Evaluation of an artifact often leads to refinements.

Research *rigor* stems from the appropriate use of the *knowledge base*. The knowledge base is formed by *foundations* used in the develop/build phase of research and *methodologies* used in the justify/evaluate phase. The knowledge base consists of previous contributions to IS research and related disciplines. *Contributions* in Design Science are assessed by their application to the identified business need in the appropriate environment.

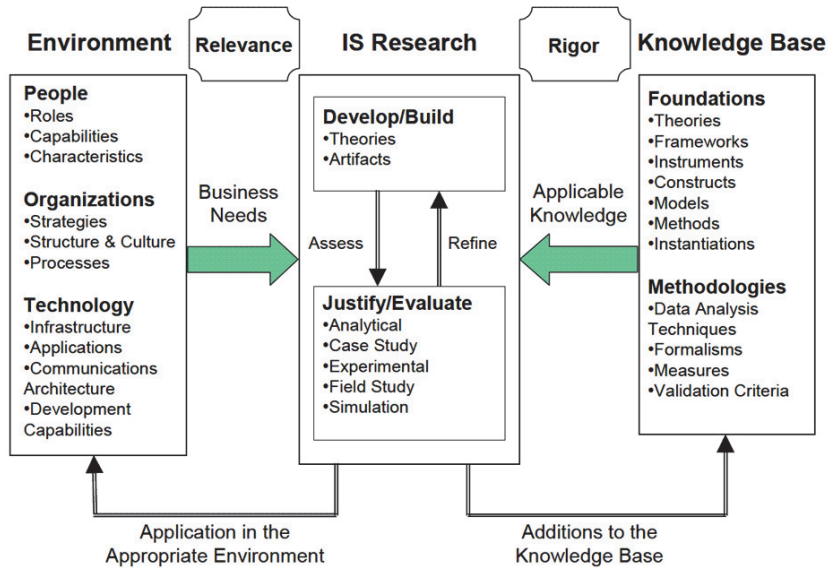


Figure 2. Information Systems Research Framework (Hevner et al., 2004, redrawn).

1.3.2 This work as Design Science research and outline of contributions

This research spans from the identification of business needs to the construction of artifacts, their evaluation, and their (limited) application in the business context. Thus it exhibits almost a full cycle of Design Science research.

Figure 3 illustrates the main areas of research of this thesis. A conceptualization of configuration knowledge was developed and served as a foundation for a developed sales configurator called WeCoTin. This formed the main trunk of this work. Additional branches of research included studies on configuration of services and recommendation support for configurable offerings.

Figure 4 presents this thesis as an instance of the IS research framework. The figure indicates the main artifacts developed, main scientific foundations, evaluation, methodologies applied, and additions to the knowledge base. The general pattern was to understand needs first, then construct artifacts, evaluate them, and, if necessary, refine the artifacts.

A complementary presentation in Figure 5 illustrates the path of research related to the main artifacts of this thesis (a configuration knowledge conceptualization and WeCoTin configurator). Arrows indicate the relations of *units of research*: the source unit of an arrow is a basis for research in the destination unit. Units portrayed with square boxes indicate Design Science artifacts; *rounded boxes* indicate other results such as understanding about the business context or requirements. The lower right corner of each unit has a numeric identifier, *unit ID*. Table 1 summarizes the units of research presented in Figure 5. Column “*Id*” refers to the unit ID in Figure 5. *Description and additional references* summarizes the contents and provides references to additional

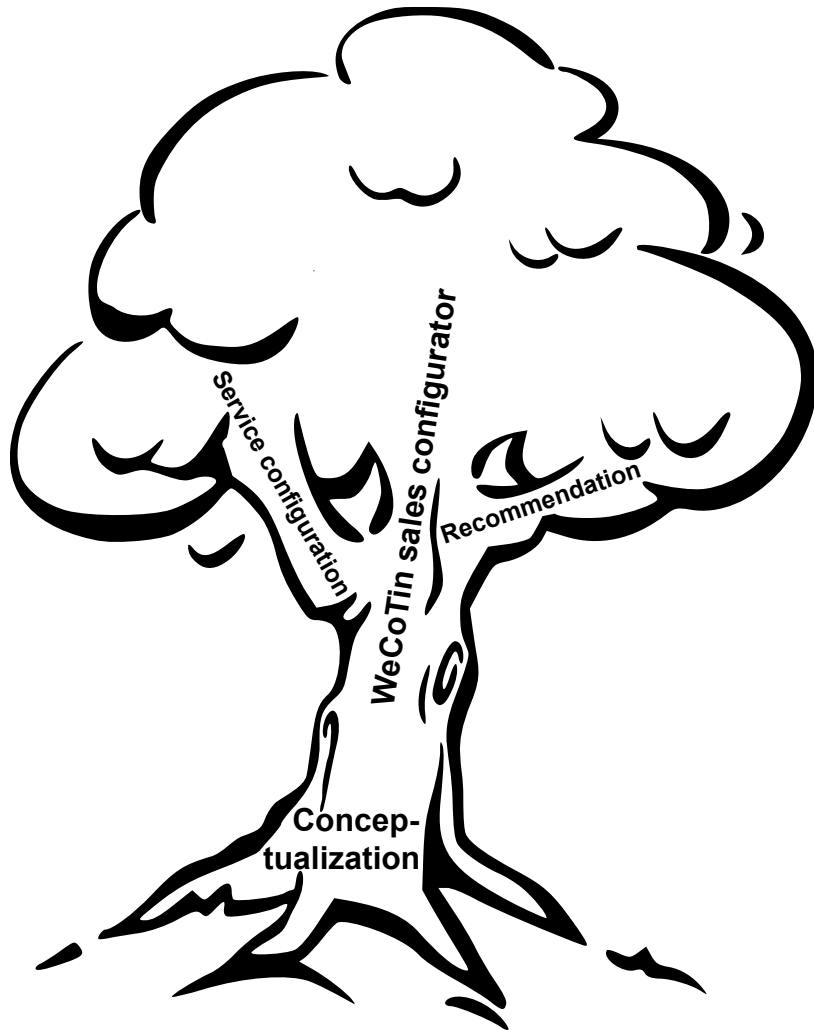


Figure 3. An overview of the research in this work. The main trunk is formed by a conceptualization of configuration knowledge and WeCoTin sales configurator. Recommendation and service configuration stem from this basis.

publications on the unit of research; *Where* identifies publication(s) annexed to this work (roman numerals) and the corresponding section(s) of this thesis.

On a higher level of abstraction, Gregor and Jones (2007) consider that the primary output of Design Science is information systems design theory (ISDT), providing general *prescriptions* for artifacts of the same type. In this spirit, a sales configurator information systems design theory (SCISDT) is proposed in Section 7.1.4 based on WeCoTin and its ingredients.

Figure 6 illustrates the additional branches of research in this thesis: configuration of services and recommendation of configurable offerings. Table 2 summarizes the units of research in relation to service configuration, and Table 3 summarizes those related to recommendation of configurable offerings.

The author of this work considers the additions to the knowledge base (Figure 4) and Design Science artifacts (the square shapes of Figure 5 and Fig-

ure 6) to be the contributions of this thesis. The main contributions are the domain-independent sales configurator WeCoTin (Unit 7 in Figure 5) and the sales configurator information systems theory (SCISDT; introduced in Section 7.1.4). Further significant contributions are the conceptualization for configuration knowledge (Unit 3) and the extended recommendation algorithms (Unit 16).

1.3.3 Case study research method

Case study research method (Yin, 2009) is applicable to examine contemporary phenomena when a researcher cannot control the events at all or control is possible only in a very limited manner. Case study is an especially suitable method for answering ‘How’ and ‘Why’ questions. ‘What’ questions can also be answered, particularly in exploratory studies. Exploratory, descriptive, and explanatory case studies are possible.

An exploratory multiple-case-study design (Yin, 2009) with four cases was applied for publication V to identify if service configuration is relevant and to characterize it in terms of offered variation and processes. The theoretical background included a previous definition of configurable products and identification of related processes (Tiihonen & Soininen, 1997a), and the classical Ws (what, when, who, where, how, and why) that have been found useful for characterizing services (Dumas, O’Sullivan, Heravizadeh, Edmond, & Ter Hofstede, 2002). Documents on existing service offering of the companies, their web pages and semi-structured interviews were applied as data sources. This provided data triangulation. Literal replication was provided by the four cases of varying background variables (see V and Section 6 for details).

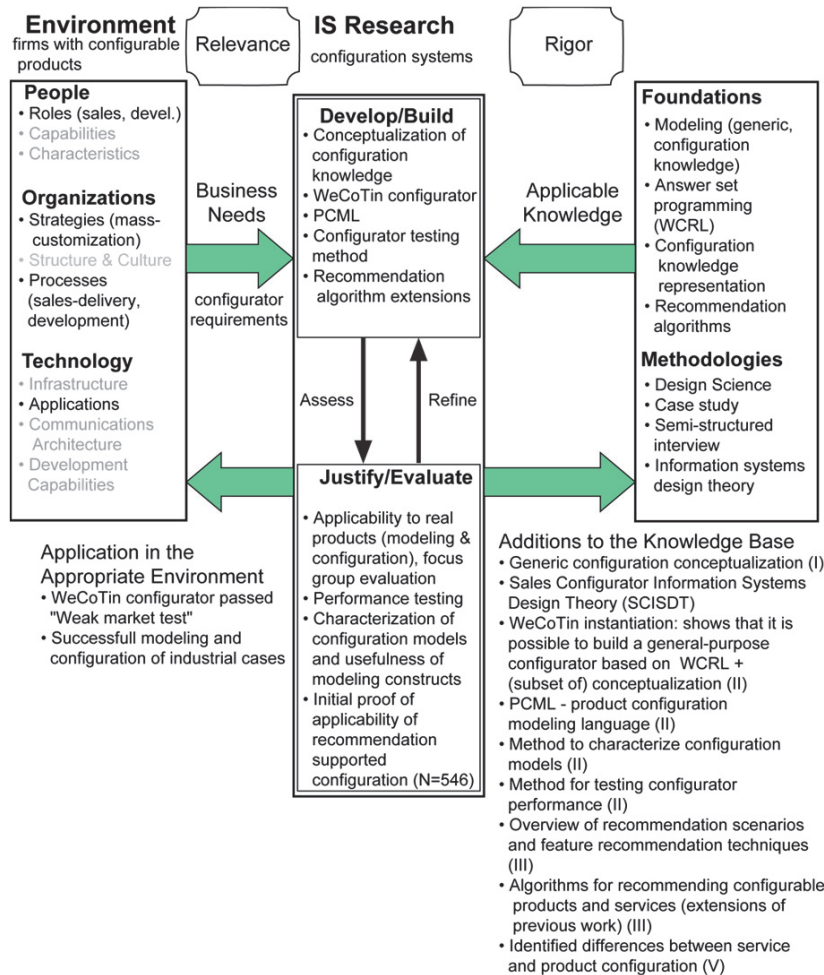


Figure 4. Instantiation of the information systems research framework in this work (adapted from Hevner et al., 2004).

1.4 Structure of the thesis

Section 2 addresses previous work and identifies gaps in the literature to justify the research questions.

Sections 3 to 6 summarize the results of publications annexed to this thesis. Section 3 summarizes a proposed conceptualization for configuration knowledge. It is based on Publication I.

Section 4 describes the main artifact constructed in this work, WeCoTin, a domain-independent configurator instantiation. Section 4 is based on publication II

Section 5 summarizes scenarios of combining recommendation technologies with configurators and the applicability of the main types of recommendation technologies. As the Design Science artifacts, extensions to previously pro-

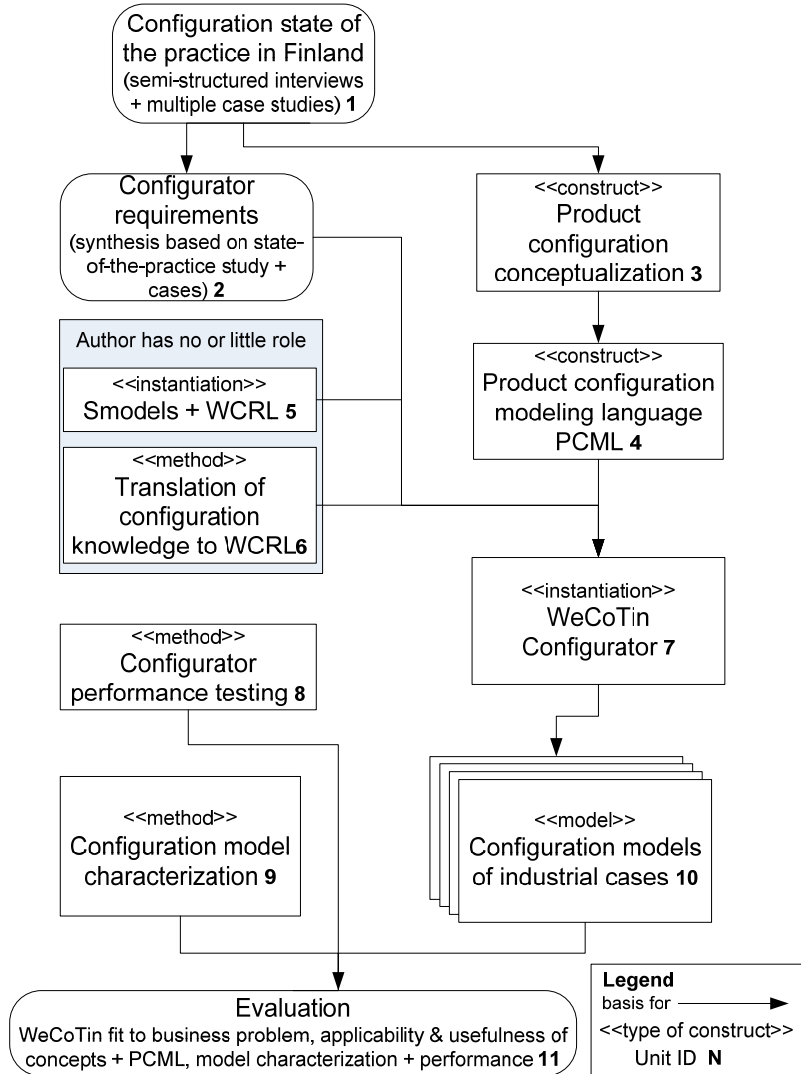


Figure 5. Research path of WeCoTin: how artifacts and evaluation are based on each other. See also Table 1.

posed recommendation algorithms are proposed. Section 5 is based on publication III.

Section 6 summarizes the relation between services and the configuration approach, based on publication V and its extended version (Tiihonen, Heiskala, Paloheimo, & Anderson, 2007).

Section 7 contains an evaluation of artifacts presented in Sections 3 to 6 based on Studies I to V. The design of WeCoTin is abstracted into sales configurator information systems theory (SCISDT).

Discussion is provided in Section 8. It includes a comparison with related work, discusses the threats of validity, answers the research questions, and identifies topics for future research. Finally, Section 9 presents conclusions.

Table 1. Summary of units of research related to WeCoTin; see also Figure 5.

ID	Description and additional references	Where
1	Understanding about the business context of configurators: configurable products, related processes, and problems. Method: semi-structured interviews in 10 companies (each about 1.5 days), construction of a single-purpose manufacturing completion configurator, case study: how to integrate a configurator into the processes of a company. (Soininen & Tiihonen, 1995; Tiihonen, 1994; Tiihonen, 1999; Tiihonen & Soininen, 1997b; Tiihonen et al., 1998)	—
2	The high-level requirements applied in the construction of the WeCoTin configurator; these are only partially documented. (Anderson & Pasanen, 2003; Tiihonen, 1994; 1999)	II, 4.1
3	A generalized conceptualization for configuration knowledge representation. (Tiihonen et al., 1998)	I, 3
4	Product Configuration Modeling Language (PCML): syntax for a subset of the conceptualization of Unit 3. (Peltonen, Tiihonen, & Anderson, 2001)	II, 4.4
5	The Smodels system provides an efficient inference engine for a weight constraint rule language (WCRL). (Simons et al., 2002; Syrjänen, 2002)	—
6	The semantics of PCML are provided by mapping of PCML to a weight constraint rule language (WCRL). (Soininen, 2000; Soininen, Niemelä, Tiihonen, & Sulonen, 2001)	—
7	The main artifact of this work: domain-independent sales configurator WeCoTin. (Tiihonen, Soininen, Niemelä, & Sulonen, 2003)	III, 4
8	A method for performance testing of configurators based on real configuration models and random requirements. (Tiihonen, Soininen, Niemelä, & Sulonen, 2002)	III, 7.1.2 p. 68
9	A method for characterizing configuration models. (Tiihonen, 2009; Tiihonen, 2010)	III, 7.1.2 p. 67
10	Applicability of WeCoTin and its modeling capabilities to industrial problems was verified by modeling and configuring of the sales view of real products and services. (Tiihonen et al., 2003)	III, 7.1.2
11	Evaluation of WeCoTin and its capabilities to model industrial problems was verified by modeling and configuring of real products and services; see Unit 10. A number of configuration models were characterized with the method of Unit 9. The run-time performance of WeCoTin was evaluated with the method of Unit 8. (Tiihonen et al., 2003; Tiihonen, 2009; Tiihonen, 2010; Tiihonen et al., 2002)	II, 7.1.2

Table 2. Summary of units of research related to service configuration. See also Figure 6.

ID	Description and additional references	Where
12	Analysis of the applicability of the configuration approach to service contract configuration and identification of special or distinguishing aspects. (Tiihonen et al., 2007)	V 6.1-6.3
13	A conceptualization for service configuration modeling, four-worlds model for configurable services (4WVM). (Heiskala, 2005; Heiskala, Tiihonen, & Soininen, 2005; Heiskala, Tiihonen, Anderson, & Soininen, 2006)	6.3
14	Service Configuration Modeling Language (SCML) for service sales configuration modeling and a translator from SCML to PCML. Based on this, WeCoTin can configure products modeled in SCML (Anderson, 2005).	6.3

Table 3. Summary of units of research related to recommendation of configurable offerings. See also Figure 6.

ID	Description and additional references	Where
15	Motivation and scenarios for applying recommendation technologies in the context of configurable offerings. Overview of the applicability of basic recommendation technologies. Overview of the feature value recommendation technologies in previous work.	III 2.5, 5.2
16	Extended versions of case-based recommendation algorithms for configurable offerings. (Tiihonen & Felfernig, 2008)	III, 5.3
17	Evaluation of the usefulness of recommendation-supported configuration. A study with 546 test users was performed. (Felfernig et al., 2010)	IV 7.1.3

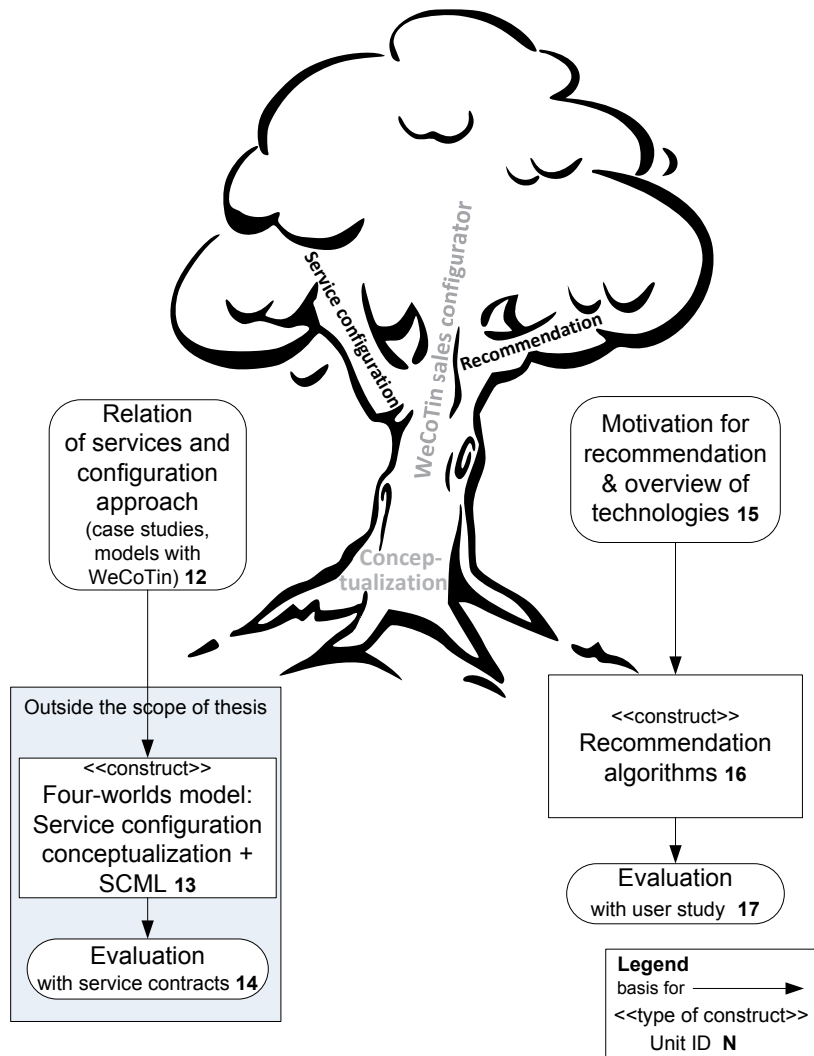


Figure 6. Research path of service configuration and recommendation of configurable offerings. See also Table 2 and Table 3.

2. Previous work

2.1 Overview of literature

Primary forums of configuration research have included special issues on configuration in the journals *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* (AI EDAM) (Darr, Klein, & McGuinness, 1998; Felfernig, Stumptner, & Tiihonen, 2011; Soininen & Stumptner, 2003) and *IEEE Intelligent Systems* (Faltings & Freuder, 1998; Sinz et al., 2007) as well as in the Configuration Workshop series arranged in 1996 and yearly since 1999 in conjunction with leading artificial-intelligence conferences.⁷ Additional forums include the *International Journal of Mass Customization* (including a special issue; (Tiihonen, Felfernig, Zanker, & Männistö, 2010), and the World Conference on Mass Customization and Personalization (MCPC) series.

Configuration has been a fruitful topic for artificial-intelligence research, including problem-solving methods, their efficient implementation, and, to a lesser extent, conceptualizations and languages for representing configuration knowledge. System instantiations based on novel approaches have been described along with their business context.

Less technical aspects, such as application of configurators in business and corresponding effects (e.g., on organization, processes, business performance), and configurator user interaction aspects are gaining momentum (e.g. Blecker, Friedrich, Kaluza, Abdelkafi, & Kreutler, 2005; Forza & Salvador, 2002a; Forza & Salvador, 2002b; Heiskala et al., 2007; Salvador & Forza, 2007). Some books guide companies on information management required by mass customization, configurator classifications, and selecting a configurator (Blecker et al., 2005; Forza & Salvador, 2006; Hvam, Mortensen, & Riis, 2008).

2.2 Basic terminology

Next, some basic terminology is defined. This enables discussion on previous work. This thesis applies some terminology of object-oriented modeling, especially the Unified Modeling Language (UML) (OMG, 2011). *Classes* have zero or more *attributes* that characterize the structure and zero or more *operations* that characterize the behavior of those objects.

⁷ The configuration workshop of 2013 was an independent event and that of 2014 was arranged in conjunction with 6th International Conference on Mass Customization and Personalization in Central Europe (MCP - CE 2014)

An *attribute* has a range of possible values (*domain*) of a value type such as integer, Boolean, string, floating point number or some object type and *multiplicity* (also known as *cardinality*): how many values an instance of the class has for the attribute. Each attribute has a name. For example, a `Car` class might have a `color` attribute that holds the color of the chassis of an instance of the `Car` class. In other words, an attribute is a mapping from a name to value(s) that characterize(s) the instance. Some attributes have fixed (constant) values, and others can be given a value.

Classes participate in *inheritance hierarchies*. *Abstract classes* cannot have any direct instances while *concrete classes* may have instances. Attributes and operations are inherited from *superclasses* and may be *redefined* or *refined* to match particular details of the class with the specializing redefinition. For example, the domain of an attribute could be restricted. In this thesis, instances of types (classes) in a configuration model are *called individuals*.

Concept (as noun) is defined in Random House Unabridged Dictionary: “1. a general notion or idea; conception. 2. an idea of something formed by mentally combining all its characteristics or particulars; a construct. 3. a directly conceived or intuited object of thought.” (Steinmetz, 1996). In this thesis, *concept* is an object of thought (meaning 3 of (Steinmetz, 1996)); concepts are also building blocks of conceptualizations.

Characteristic is a ‘distinguishing feature or quality’ (Steinmetz, 1996) of an object (e.g., concept or idea).⁸ *Property* is a characteristic or attribute of any object. More specifically, in the context of configuration modeling, configuration types (see publication I, Section 3), may define attributes, compositional structure, possibilities of participating in relationships, and some other aspects. All these are properties.

2.3 Configuration knowledge modeling

Configuration knowledge modeling offers ways to represent configuration models, requirements, and configurations. It has received significant attention in the literature.

Three primary types of configuration modeling conceptualizations can be identified. The first type is actually not a conceptualization. It is based on the idea that *configuration knowledge can be directly encoded in the presentation mechanisms of the problem-solving method*. These methods and ways to represent configuration knowledge are outlined in the context of problem solving in configurators (Section 2.4.2).

The second type is *configuration-domain-specific conceptualizations*, which are independent of problem-solving methods. These can be roughly classified as *connection-based* (Mittal & Frayman, 1989), *resource-based* (Heinrich & Jüngst, 1991), *structure-based* (e.g., Cunis et al., 1989), or *function-based* (Najmann & Stein, 1992) approaches. The conceptualizations have little in

⁸ Terms ‘characteristic’ and ‘property’ are defined with various meanings in different streams of literature. For example, the domain theory of (mechanical) engineering design (Hansen & Andreasen, 2002) defines meanings that should not be confused with those of this work.

common, other than the central notion of a *component*. Term *component* requires clarification and it is used in this work in two senses. First, without qualification, component is used in the general sense: “a constituent part; element; ingredient” (Steinmetz, 1996). Second, components are building blocks of products in the sense that products (product individuals) consist of components (component individuals). The nature of components is discussed in Section 2.3.1 below.

The third and the most recent type of conceptualization includes unified approaches that combine the ideas of the individual approaches into a covering ontology or conceptualization. These are discussed in the context of related work; see Section 8.1.1 (p. 81).

2.3.1 Component

As stated above, literature identifies components as building blocks of products in the sense that products (product individuals) consist of components (component individuals). This is a common notion, but the exact nature of components is slightly ambiguous and depends on the adopted approach to configuration knowledge modeling and also on the views of individual authors.

According to many authors (including the one of this thesis), pre-designed components may have specification variables a.k.a. *parameters*, the values of which need to be specified to manufacture or configure individual components, e.g. physical dimensions, surface material, color, or capacity (Aldanondo, Hadj-Hamou, Moynard, & Lamothe, 2003; Fleischanderl et al., 1998; Tiihonen & Soinen, 1996; Wielinga & Schreiber, 1997). Many components are non-parametric – components do not have any specification variables; they can be manufactured by identifying the component only; pre-existing drawings, product structures, and other information are available due to pre-design. Some authors only recognize non-parametric components (e.g., Mittal & Frayman, 1989). All components – parametric or not – may be characterized with properties such as weight, material, capacity or power. The value of some properties is constant, and the value of some properties (e.g. weight) may depend on selected parameter values. Attributes can be used both to represent parameters of components, and to represent derived or fixed properties. Attributes can be necessary or optional – an optional attribute does not have to have a value in a complete configuration, while a necessary attribute needs a value.

Structure-based configuration modeling has emphasis on the modeling of the compositional structure of products (e.g., Cunis et al., 1989). Often a component is a physical, usually separable part of another component. In the configuration, domain this *whole-part relationship* is usually called *has-part relationship*. The relationship is irreflexive, antisymmetric, and transitive (Artale, Franconi, Guarino, & Pazzi, 1996; Štorga, Andreasen, & Marjanović, 2010). Thus, a component cannot be a part of itself (irreflexivity). The antisymmetric nature implies that if W has-part P , it does not hold that P has-part W . Transitivity means that if W has-part P and P has-part Q , also W has-part Q holds.

A central phenomenon in the context of configuration is determining which components (can) become part(s) of a whole. For example, a *Car* is to have exactly one *Motor* that can be selected out of available motor types e.g. *Motor_1.8_120HP_Petrol* or *2.0_100HP_Diesel*. Here, several aspects are noteworthy. First, it is desirable to be able to refer to parts by their *generic part name* (Artale et al., 1996), for example *Motor*. Second, *cardinality* indicates a valid number of parts with a generic part name. For example, a *Car* might have exactly one *Motor*, exactly four *Wheels*, and an optional (0 or 1) *Sunroof*. Third, only some types of component (*allowed types, possible part types*) are eligible for a specific part name; in this example *Motor_1.8_120HP_Petrol* or *2.0_100HP_Diesel* could be eligible while *Motor_1.8_180HP_Turbo_Petrol* might not be. Other aspects of the part-whole relationship are discussed in Section 3 and publication I.

A major aspect of a component in the configuration domain is connectivity: the possibility or requirement to connect a component with other components. This is the basis of *connection-based* approaches to configuration knowledge modeling (Mittal & Frayman, 1989), *Ports* represent connection interfaces; different port types imply different connections. Compatibility of components can be modeled with ports: if ports allow components to be connected, they are compatible. In a pure connection-based approach, the compositional structure is not explicitly modeled with specific concepts. Rather, ports may be applied to connect wholes and their parts (e.g., Felfernig, Friedrich, & Jannach, 2000b).

A further notion is that components can be modeled via *resources* that model the production and use of some entity, such as power or expansion slots (Heinrich & Jüngst, 1991; Heinrich & Jüngst, 1996). The underlying idea of this *resource-based* configuration modeling is that some component individuals produce a resource and other component individuals use it. There must be enough production to cover use. Pure resource-based configuration modeling characterizes components are only by their resource production and use.

In natural language, discussion on components does not explicitly distinguish between configuration model knowledge and configuration solution knowledge. For example, the sentence “Car has an engine as a part” can be interpreted in two ways. As configuration model knowledge, the sentence can be understood as saying that every car individual must have an engine individual as a part. As configuration solution knowledge, it states that a configuration includes a car individual that has an engine individual as a part. To make this distinction explicit, *component type* and *component individual* are presented as central concepts in a configuration conceptualization discussed in Section 3 and publication I.

2.3.2 Function and feature

According to the ‘domain theory’ of mechanical artefact design, any product is to support a *transformation process* where the interplay between an *operator* (human) and the artefact delivers *effects* that are necessary for the (often

stepwise) transformation of an *operand* (Hansen & Andreasen, 2002). The end state of the operand fulfils the purpose or satisfies the intended human need. The operand can be material, energy, data, or biological objects. Additional types of characterizations include ‘universal virtues’ that include cost, quality, time, efficiency, flexibility, risk, and environmental effects (Hansen & Andreasen, 2002). This thesis adopts the view of publication I, where the functional view of products consists of characterizations of the product that a customer or sales person would utilize to describe the products. Many of the functional characterizations (*functions*) concern the purpose-oriented transformations, related universal virtues or *quality attributes* such as reliability, usability, security, or availability. The conceptualization of Section 3 and publication I introduces concepts *function type* and *function individual* to represent the functional view in configuration models and configurations, respectively.

The authors of publication II consider *feature* as a generalization of function; some aspects of the sales view of configurable products are more adequately called ‘features’ than ‘functions.’ Examples include the type of bed end of a hospital bed, the configurable method of sweeping a fireplace, or the desired shape of an extension element of a fireplace. Furthermore, the authors consider that ‘feature’ is closer to natural language as a better match than ‘function’ to describe the sales view configurable products. Therefore, the configuration modeling language of WeCoTin configurator is based on concepts *feature type* and *feature individual* (Section 4).

Literature on recommender systems considers that products (‘items’) are characterized by *features*. In this context, features can often be thought of as attributes. For example, the feature `color` of a personal computer `PC1` might have value `red`, and another computer `PC2` might have value `color = black`.

It is commonly recognized that features or functions are implemented or realized by component individuals (e.g., Aldanondo, Rouge, & Véron, 2000; Sabin & Weigel, 1998). With respect to domain theory (Hansen & Andreasen, 2002), features or functions roughly correspond to the transformation domain that focuses on the purpose-oriented transformation of the operand. Component individuals in a configuration correspond to the part domain that specifies the parts that can be produced and assembled into a functioning product. The organ domain with mechanical product's active elements is omitted.

According to some authors, the sales configuration process of highly customizable products takes entirely place in the functional view (Aldanondo et al., 2000). Najmann & Stein (1992) proposed that objects (components) be characterized by their (attribute-like) functionalities; demands (user requirements) are also specified in terms of functionalities. The construct resembles resource-oriented configuration modeling.

Configuration knowledge often includes *constraints* for specifying the interdependencies of entities such as component individuals or function individuals or both. A constraint is a formal rule, logical or mathematical or a mixture of these, specifying a condition that must hold in a correct configuration. Literature often mentions *requires* and *incompatibility* constraints (Felfernig, Friedrich, Jannach, & Zanker, 2002; Felfernig, 2007; Tiihonen & Soinen, 1996).

A requires-constraint states that some component(s) or parameter value(s) require other component(s) or parameter value(s) to function. For example, a sound recording unit requires a microphone to function. Incompatibility-constraint states that some component(s) or parameter value(s) cannot be used together. For example, in the context of a car, a combination of an air conditioning system and an automatic gearbox is incompatible with a low power engine.

Research Question RQ1: What are the concepts central to configuration knowledge?

2.4 Configurators

2.4.1 Configurators—an extensively researched topic

Numerous configurators have been developed both as research prototypes and as commercial software. The landmark R1/XCON was deployed at Digital Equipment Corporation in the early 1980s (McDermott, 1982), and experiences, benefits, and challenges of using it have been widely documented; see, e.g., Barker, O'Connor, Bachant & Soloway (1989), Sviokla (1990), and McDermott (1993).

Major research efforts have been devoted to configurators applicable to solving general configuration tasks instead of a specific domain. These include COSSACK (Frayman & Mittal, 1987), PLAKON (Cunis et al., 1989; Cunis, Günter, & Strecker, 1991) and its successor KONWERK (Günter & Hotz, 1999; Hotz & Günter, 2014), and COCOS (Stumptner et al., 1994).

A large number of commercial general-purpose configurators exist. Trilogy SalesBUILDER (Hales, 1992) was among the first. ILOG offered a generic configuration engine to be used in other vendors' systems (Junker & Mailharro, 2003a; Mailharro, 1998). Anderson (2005) identified 30 vendors by their Web pages. In addition, prominent enterprise resource planning system and CRM vendors have one or more configurators, e.g., SAP⁹ and Oracle.¹⁰ Furthermore, configuration capabilities are common in Product Lifecycle Management systems, e.g., Dassault Enovia (3DSEnovia, 2012), Siemens Teamcenter (Siemens, 2011), and PTC Windchill (PTC, 2012).

Configurators are deployed relatively widely. For example, the International Configurator Database listed 970 Web-based configurator instances that are available for customers of corresponding companies (cyLEDGE, 2013). Some of these may be single-purpose “hard-coded” systems, while others are built on general-purpose configurators that are of interest in this work.

⁹ SAP R/3 “variant configurator” and “IPC, Internet Pricing and Configuration” (Haag, 2005; Haag, 1998)

¹⁰ Oracle Configurator (Damiani et al., 2001; Oracle, 2004; Oracle, 2009), JD Edwards EnterpriseOne (Oracle, 2012), PeopleSoft Enterprise Configurator (Oracle, 2005), and Siebel Configurator (Oracle, 2007). Oracle has developed ‘Fusion Configurator Engine’ (Sawtelle, 2010), applied in the Oracle Configurator.

Both numerous individual configurator instantiations and general-purpose configurators that enable the creation of such instantiations exist. In consequence, the author of this work holds that developing artifacts in these categories is not a scientific contribution as such; greater novelty or deeper principles are required.

2.4.2 Problem solving in configurators

Numerous problem-solving methods have been applied to configuration tasks; several overviews of the topic exist. At least rule-based approaches, constraint satisfaction and its dynamic extensions, several logic-based approaches, and different formalisms of propose-and-revise methods have been applied; for summaries, see Stumptner (1997) and Sabin and Weigel (1998). Of these methods, constraint satisfaction is the most widely applied.

In their taxonomy of types of problem-solving methods for design and configuration, Wielinga and Schreiber (1997) consider *configuration problem-solving methods* a subtype of *design methods*. Configuration problem-solving methods can be further divided into *knowledge-intensive methods* and *uniform methods*. Uniform methods apply the same reasoning methods to all problems, whereas knowledge-intensive methods use (explicitly modeled) knowledge to constrain and direct problem solving. Knowledge-intensive methods (*propose, critique, and modify; case based, and hierarchical*) are not considered further in this work: the author considers uniform methods to already be mature enough for supporting the configuration tasks in sales configuration of many products and services.

Uniform methods include *constraint solving* and *logic-based methods*. Constraint satisfaction (CSP) and its extensions have gained significant popularity (Fleischanderl et al., 1998; Mailharro, 1998; Mittal & Falkenhainer, 1990). Many authors, e.g., Desisto (2004) and Haag, Junker & O’Sullivan (2007),¹¹ consider constraint-based methods ideal for solving configuration problems. Constraint-based methods can be extended with preference programming. Here, the idea is to express preferences and to provide inference that supports finding solutions that maximally satisfy preferences in such a way that more important preferences are satisfied before less important ones (Junker & Mailharro, 2003b).

A constraint satisfaction problem is a tuple (V, D, C) . Here, V is a set of finite domain variables, $V = \{v_0, v_1, \dots, v_n\}$. Each variable has a (usually finite) domain that specifies the possible values of the variable, and the set of domains is D , $D = \{dom_0, dom_1, \dots, dom_n\}$. C is a set of constraints specifying restrictions on the allowed combinations of variable value assignments. A solution to a constraint satisfaction problem is a set of assignments to each variable $\{v_0 = x_0, v_1 = x_1, \dots, v_n = x_n\}$ such that each $x_i \in dom_i$ and the assignments are consistent with the set of constraints C (Mackworth & Freuder, 1985). When representing a configuration problem, each part to be selected and each

¹¹ An essay in (Sinz et al., 2007) that is based on the Configuration Workshop of the 17th European Conference on Artificial Intelligence (ECAI 2006).

configurable attribute is represented as a variable, and the domain of the variable is defined to include the respective possible values. There are two subsets of constraints $C = C_{CM} \cup C_{Req}$. C_{CM} specifies configuration model relations between allowed values, e.g., “incompatibility” and “requires” constraints between components or their attribute values. C_{Req} specifies the set of customer requirements. A challenge for modeling with basic CSP is that many configuration problems are dynamic in the sense that when, e.g., a component is selected into a configuration, some parameters, parts or connections related to that component need to be determined, leading to a need to introduce new variables or constraints. Dynamic CSP (DCSP; (Mittal & Falkenhainer, 1990; Soininen & Gelle, 1999) and Generative CSPs (Fleischanderl et al., 1998; Stumptner et al., 1998) address this problem.

Several logic-based methods have been applied to solve configuration problems successfully. These include direct programming in Prolog or through a higher-level modeling layer (e.g., Searls & Norton, 1990). Description logics (e.g., Baader, 2009) have been applied (McGuinness & Wright, 1998b; Wright et al., 1993; Wright, McGuinness, Foster, & Vesonder, 1995). Constraint logic programming has also been applied (Sharma & Colomb, 1998). Answer set programming (ASP) makes it possible to express the problem as a theory consisting of logic program rules with clear declarative semantics, and the stable models of the theory correspond to the solutions (answer sets) to the problem (Simons et al., 2002). The theories are expressed in weight constraint rule language (WCRL). WCRL is equipped with weight constraints for representing weighted choices with lower and upper bounds and with conditional literals restricted by domain predicates to encode the sets of atoms over which the choices are made. Weight constraint rules have been applied directly to model configuration (Schenner, Falkner, Ryabokon, & Friedrich, 2013; Syrjänen, 2000) and reconfiguration (Friedrich, Ryabokon, Haselböck, Schenner, & Schreiner, 2011; Schenner et al., 2013) problems in research systems. Furthermore, a method has been proposed to translate configuration domain modeling concepts into weight constraint rules (Soininen, 2000; Soininen et al., 2001). Following this idea, an experimental system, OOASP, showed the feasibility of checking a configuration, completing a configuration, and performing reconfiguration (Schenner et al., 2013).

Sometimes different problem-solving methods have been combined, such as description logic with constraint satisfaction (Junker & Mailharro, 2003a).

Previous work left room for a configurator that would be based on high-level modeling conceptualization and the idea of translation of configuration knowledge into weight constraint rules, a form of logic programs (Soininen, 2000; Soininen et al., 2001).

Research Question RQ2: How to construct a practical and computationally well-founded sales configurator?

2.5 Recommendation of configurable offerings

Recommender systems support users in selecting relevant *items* (e.g., books, movies, insurance policies) in cases in which they do not have sufficient personal experience of the alternatives (e.g., Resnick & Varian, 1997); Burke, (2002). In the context of recommender systems, the term *feature* is often used to refer to a distinguishing characteristic of an item. Here, features can be thought of as attributes. For example, the feature `color` of a personal computer `PC1` might have value `red`, and another computer `PC2` might have value `color = black`.

Recommender systems have been relatively widely applied to recommend simple products such as books and movies. For example, Amazon.com applies these technologies widely (e.g., Linden, Smith, & York, 2003). However, the capability to support configuration tasks with recommendation technologies is still in its infancy.

Next, basic recommendation technologies are outlined, followed by motivation and identification of research gaps in the context of configurable offerings.

2.5.1 Basic recommendation technologies

An *active user* is a user whose decision making the system currently supports, typically in a personalized way. We omit discussion about non-personalized approaches such as setting static defaults in configuration models.

Collaborative filtering (Adomavicius & Tuzhilin, 2005; Konstan et al., 1997) is one of the most commonly used recommendation technologies. It provides recommendations based on the opinions of the active user and other users (e.g., ratings or purchasing data). The basic idea is to identify users who are similar to the active user and to recommend their highly rated items that are unknown to the active user. A *similarity function* determines the similarity of opinions about items to calculate *nearest neighbors*, which are users with similar preferences.

Content-based filtering (e.g., Pazzani, 1999) recommends items similar to those which the active user has preferred in the past. Items are described by a number of *keywords* or *features*. A *user model* contains previous opinions about items; these are often presented as keywords or features as well. A similarity function is used to calculate *nearest neighbors*, which are in this case those items with the highest similarity compared with the preference information given in the user profile (Burke, 2000). The approach is typically applied for recommending text-based items such as articles or Web pages.

Utility-based recommendation estimates the relative satisfaction or desirability of consumption of an item, i.e., *utility* for the customer, and recommends items with the highest utility. Here, multi-attribute utility theory (MAUT) (Dyer, 2005; Von Winterfeldt & Edwards, 1986) is exploited in its additive form. Domain-specific *interest dimensions* are identified. For personal computers, interest dimensions could be economy, reliability, graphics performance and weight. Items are given numeric utility values with respect to each interest dimension. The user specifies his preferences in terms of im-

portance (weight) of each interest dimension. Given this information, item utilities can be computed for the active user by summing up the weights multiplied by the utilities of the interest dimensions.

Compromise-driven retrieval (McSherry, 2003) recommends items that are characterized by attributes. The active user specifies the desirable values of some attributes, and the system retrieves items that are similar. In McSherry's terms, similarity is defined slightly nonintuitively, based on the idea that most users would like to maximize or minimize the values of many numeric product attributes. These attributes are denoted as *more-is-better* (e.g., resolution or optical zoom ratio of digital cameras) or *less-is-better* (e.g., price or weight). Furthermore, for *nearer-is-better* attributes, users prefer a high similarity between their preferences and the corresponding product. Utility of a product is the sum of weighted attribute similarity values. McSherry aims to support making compromises by identifying and presenting only cases that have similar compromises, hence the name *compromise-driven retrieval*.

Knowledge-based recommenders exploit explicit information about items and user requirements and how they can be satisfied (Burke, 2000; Felfernig, Isak, Szabo, & Zachar, 2007). *Constraint-based recommendation* (Felfernig & Burke, 2008) is a knowledge-based approach wherein alternative items and potential customer requirements are described on the basis of a set of features and the corresponding constraints. Felfernig et al. (2007) describe a system in which *filter constraints* match customer requirements to suitable items. *Compatibility constraints* ensure the consistency of requirements. In addition, explanation and repair functionalities are provided to support the user in resolving inconsistencies.

Case-based recommendation (Burke, 2000) is another type of knowledge-based recommendation. In contrast to content-based filtering and collaborative filtering, elementary properties of items (e.g., PC price, hard-disk size) from the previous recommendation sessions are taken into account rather than extracted keywords, categories, or the identity of the user. Case-based reasoning exploits similarity functions and *Bayes predictors* on previous sessions to determine interesting items and feature settings fitting the wishes and needs of users. Bayes predictors allow the prediction of interesting items on the basis of their probability of being selected given the existing user preferences. *Naïve Bayes predictors* assume that variables are independent, which makes them computationally more feasible but potentially less accurate than Bayes predictors that take into account dependencies between variables. Naïve Bayes predictors have been applied for content-based filtering (Pazzani, 1999) and case-based recommendation (Cöster, Gustavsson, Olsson, & Rudström, 2002).

For configuration purposes, Geneste and Ruet (2001) proposed a case-based approach: identify a similar configuration (case) and adapt it to solve the current configuration task. Cöster et al. (2002) proposed three case-based algorithms for recommending configurable offerings. Two of them are based on naïve Bayes predictors: the *naïve Bayes voter* recommends individual feature values, and the *most popular choice* recommends feature values for the set of

features for which the user does not have a value, typically to complete the configuration. The *weighted majority voter* recommends individual feature values. It is computationally and conceptually simpler than naïve Bayes voter and most popular choice.

Hybrid approaches attempt to combine the benefits of different approaches to provide better recommendations, and challenges may also follow (Adomavicius & Tuzhilin, 2005; Burke, 2002).

2.5.2 Recommendation of configurable offerings

As Aldanondo, Véron, & Fargier (1999) recognized, a configurator usually has to work in a situation in which a customer is looking for a solution, typically without explicitly formulated needs and with a varying level of knowledge about the possible solutions. It is the task of the supplier (possibly through a configurator) to understand the customer's need and to find a product that meets that need. However, customers usually do not know their (detailed) preferences beforehand; preferences are constructed (Bettman, Luce, & Payne, 1998; Häubl & Murray, 2003) within the scope of a configuration session.¹²

Challenges of choice navigation may hinder the sales success of configurable offerings. Especially nonexpert users who configure products or services may be overwhelmed by the offered set of alternatives. In such a situation, a user may become dissatisfied or decide against making a choice—a phenomenon known as *mass confusion* (Huffman & Kahn, 1998).

An approach to reduce potential for mass confusion is to provide personalized recommendations for individual selections or for completing a configuration (Ardissono et al., 2002; Ardissono et al., 2003; Cöster et al., 2002; Stegmann, Leckner, Koch, & Schlichter, 2006; Stegmann, Koch, Lacher, Leckner, & Renneberg, 2003).

Numerous authors (e.g., Falkner, Felfernig & Haag, (2011); Resnick & Varian, (1997); Stegmann et al., (2006)) have proposed excluding (filtering out) some features (e.g., attributes) or alternatives (e.g., attribute values) that could be determined uninteresting from the customer's point of view. A similar idea is to recommend features (configuration decisions) to configure next based on the estimated interest of the user (Falkner et al., 2011; Felfernig & Burke, 2008; Wang & Tseng, 2011).

Yet another approach is to provide personalized preconfigured packages (proposals for complete configurations), e.g., in the context of tourism packages (Zanker, Aschinger, & Jessenitschnig, 2007).

Use of recommendation in the context of configurable products and services such as financial services, personal computers, or cars is relatively limited, and few commercial configurators apply recommendation technologies; despite some efforts we were unable to find literature showing substantial deployments. An exception (with limited configuration functionality) was (Felfernig et al., 2007).

¹² Of course, some customers may know in detail both their needs and the available offering. For such users, recommendation support may not provide significant benefits.

To sum up, the author sees the need for the recommendation of configurable offerings, but the proposed approaches have not been used widely, and even “toy examples” are relatively few. Therefore, we ask:

Research Question RQ3: Can users be effectively supported in finding suitable products and services with personalized recommendations?

2.6 Mass customization and configuration of services

2.6.1 Services—a minimal overview

Despite a consensus on the importance of services, there is no consensus on the exact definition. In this work,¹³ we apply the following definition “reluctantly” proposed by Grönroos (2007): “A service is a process consisting of a series of more or less intangible activities that normally, but not necessarily always, take place in interactions between the customer and service employees and/or physical resources or goods and/or systems of the service provider, which are provided as solutions to customer problems” (p. 52). Prominent service definitions consider processes central to the nature of services (Fitzsimmons & Fitzsimmons, 2004; Grönroos, 2007; Vargo & Lusch, 2004a).

Services are often attributed with characteristics that differentiate them from tangible goods: *intangibility*, *heterogeneity*, the *inseparability* of production and consumption, and *perishability*, (collectively known as the *IHIP* characteristics; (Grönroos, 2007; Zeithaml, Parasuraman, & Berry, 1985). Another commonly mentioned service characteristic is *no change of ownership*; service transactions do not usually result in the change of ownership (Lovelock & Gummesson, 2004). But not all of these characteristics apply in all service contexts, and the usefulness and general correctness of these characteristics are being debated (Edvardsson, Gustafsson, & Roos, 2005; Lovelock & Gummesson, 2004; Vargo & Lusch, 2004b).

The field of services is extremely diverse. Therefore numerous service typologies have been presented (Chesbrough & Spohrer, 2006; Cook, Goh, & Chung, 1999; Edvardsson et al., 2005; Lovelock, 1983; Silvestro, Fitzgerald, Johnston, & Voss, 1992; Zeithaml et al., 1985). For example, Cook et al. (1999) listed about 40 classification schemes of services presented in previous work.

Offerings usually consist of a mix of tangible goods and services. In other words, there is a spectrum from pure tangible goods to pure services (Kotler, 1988; Levitt, 1981). Services are not only provided in the traditional service sector. Manufacturers of goods offer invoiced or bundled services such as information, logistics, software, and upgrades and *hidden services* such as invoicing and complaint handling (Grönroos, 2007).

¹³ The scope of this work excludes Web services, “the computing paradigm that utilizes services as fundamental elements for developing applications” (Papazoglou & Georgakopoulos, 2003, p. 25).

2.6.2 Mass customization and configuration of services

Mass customization of services can potentially provide both good fit with customer needs and the benefits of standardization (Bowen & Youngdahl, 1998; Da Silveira, Borenstein, & Fogliatto, 2001; Gilmore & Pine, 1997; Hart, 1995; Harvey, Lefebvre, & Lefebvre, 1997; McLaughlin, 1996; Pine, 1993; Sundbo, 2002). However, explicit research on service mass customization is relatively sparse, and a need for more research has been identified (Beckett, 1996; Da Silveira et al., 2001; Duray, Ward, Milligan, & Berry, 2000; Heiskala et al., 2007; McLaughlin, 1996). Lately, there has been significant interest in mass customization of services. For example, the MCPC 2007 conference included a session on service mass customization, and the proceedings of the IMCM & PETO conference in 2008 were titled “Mass Customization Services” (Edwards, Blecker, Salvador, Hvam, & Friedrich, 2008).

Results of research on goods mass customization may not be directly applicable to services, because of the characteristics of services (Da Silveira et al., 2001; Harvey et al., 1997). On the other hand, products and services are often discussed without much differentiation between services and physical products (see, e.g., Gilmore & Pine, 1997; Hart, 1995; Pine, 1993).

These differing views raise a question: is configuration of services business as usual, or is there anything that distinguishes configuration of services from that of physical goods?

Specific literature on the relation between configuration approach and services is scant (Heiskala, Paloheimo, & Tiihonen, 2005). Previous work has suggested that services are amenable to the configuration approach: The idea of composing services according to customer requirements from pre-designed service modules or components to achieve a good fit to customer requirements and efficiency (of mass customization) has been recognized by several authors (Ardissono et al., 2003; Baida, Akkermans, & Gordijn, 2003; Böhmman, Junginger, & Krcmar, 2003; Dausch & Hsu, 2006; McLaughlin, 1996; Meier & Massberg, 2004; Meyer & DeTore, 2001; Stolze & Field, 2000; Sundbo, 1994). Such composition effectively suggests service configuration. The configuration approach to mass customizing services has been applied or proposed in several industries. These include financial services (Felfernig, 2007; Haag, 2008; Junker & Mailharro, 2003b), telecommunications services (Oracle, 2004; 2009; SAP, 2001; 2005), travel services (Goy & Magro, 2004a; Goy & Magro, 2004b; Werthner & Ricci, 2004), and maintenance services of industrial goods (Dausch & Hsu, 2003; 2006; Meier & Massberg, 2004).

Configurators developed primarily for goods are indicated to support configuration of services. Twenty of 30 commercial configurator vendors studied by Anderson (2005) indicated that their configurators supported services. Only two vendors described their modeling concepts, and neither introduced any service-specific concepts. No modeling examples were found. Still, major business software vendors such as Oracle and SAP provide functionality at least for configuring telecommunications services (Oracle, 2004; 2006; SAP, 2001; 2005). It is evident that *financial services* are also configured (Anderson, 2005; Felfernig et al., 2007; Haag, 2008).

To sum up, we were unable to find adequate overviews on service configuration to understand whether service configuration differs from the configuration of physical products. Questions such as the following remained without answers:

- Are processes related to configurable services identical to those for physical products? If not, what are the differences?
- What variation is offered in the context of configurable services?
- What is the nature of components or modules of services?
- Do service characteristics such as IHIP affect service configuration?
- What kinds of services are amenable to mass customization by configuration?
- Can configurators designed for physical products be applied to manage offered variation of services?

Research Question RQ4: How does service configuration differ from the configuration of physical products?

By now, the research questions have been identified. The following Sections 3 to 6 summarize the corresponding results that are evaluated in Section 7 and discussed in Section 8. Finally, Section 9 presents conclusions. A more detailed overview of the structure of this work was provided in Section 1.4 (see p. 19).

3. Conceptualization for configuration knowledge

This section summarizes a conceptualization for configuration knowledge that synthesizes and extends earlier approaches (I). The *conceptualization* defines objects that are of primary interest in configuration knowledge as well as the key relations among these objects. We prefer to call it a conceptualization because there is no detailed and explicit formalization that would be required for it to be called *ontology* (Gruber, 1993).

The basic structure of the conceptualization is presented in Figure 7. Please note the example in Section 3.8 (Figure 8, p. 42). A set of *configuration model concepts* forms the top-level taxonomy. Concepts and classes defined at this (meta)level are applied to represent configuration models. This *configuration model knowledge* specifies the entities that can appear in a configuration, their properties, and the rules on how the entities and their properties can be combined. Individuals (instances) of configuration model concepts describe individual *configurations* and thus represent *configuration solution knowledge*. Finally, *requirements knowledge* specifies the systematized requirements on the configuration to be constructed. Requirements knowledge can be specified with the same concepts as configuration model knowledge and configuration solution knowledge, although it plays a different role in problem solving.

The conceptualization covers *connection-based* (Mittal & Frayman, 1989), *resource-based* (Heinrich & Jüngst, 1991), *structure-based* (e.g., Cunis et al., 1989), and *function-based* (Najmann & Stein, 1992) approaches presented in the literature. It integrates the previous main approaches to configuration knowledge modeling while treating the main concepts uniformly with respect to several criteria.

3.1 Types, individuals, and classification

Types and *individuals* clearly distinguish between the entities that occur in configuration model knowledge and configuration solution knowledge. A configuration can contain individuals of subtypes of the following main types of configuration model knowledge: *component*, *port*, *resource*, and *function*.

The main types are organized in a *classification hierarchy* in the usual manner (e.g. Rumbaugh, Blaha, Premerlani, Eddy, & Lorensen, 1991). A type has a

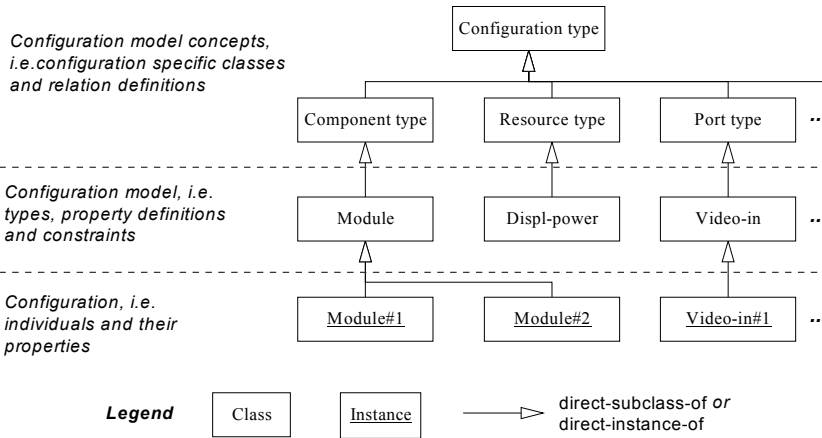


Figure 7. Basic structure of the conceptualization (I).

set of *property definitions* such as attribute, part, and port definitions. A type *inherits* the properties of its *supertypes* in the classification hierarchy. When a type inherits definitions from a supertype, the type can use the inherited data “as such,” or it can “modify” the inherited data by means of *refinement*. Refinement is semantically based on the notion that the set of potential valid individuals directly of the subtype is smaller than the set of valid individuals directly of the supertype.

A type is either *abstract* or *concrete*. *Abstract types* are used as supertypes in a taxonomy that enables gathering common knowledge related to their subtypes. Only individuals that are directly of a *concrete type* can occur in a complete configuration as they are accurate enough to be used in an unambiguous configuration.

A component type is either *dependent* or *independent*. A valid component individual that is directly of an independent component type may exist in a configuration without being a part of something, whereas a valid component individual that is directly of a dependent type may not.

3.2 Attributes

Component, port, resource, and function types can define *ATTRIBUTES*. These represent the parameters, derived or fixed properties of interest that are to be represented as the variables or constants of an individual of the type. Attributes have a name and a value type. Value types can be formed from basic value types Boolean, integer, string, or float. Some attributes have fixed *values*, and others can be given a value. Attributes can be necessary or optional—an optional attribute does not have to have a value in a complete configuration, whereas a necessary attribute must have a value.

Examples of attributes include the physical dimensions of parametric component types, surface material, color, resistance, and capacity.

3.3 Component types and compositional structure—parts

A *component type* represents a distinguishable entity in a product that is meaningful to product configuration in the sense that a configuration is composed of *component individuals* of their respective component types.

The compositional structure is important for configuration because products are commonly described in terms of their structure for design, manufacturing, or maintenance purposes. The conceptualization directly supports generalized product structures with a varying number of mandatory, optional, and alternative parts via *PART DEFINITIONS* that can be specified in *component types* and *function types*. The semantics of a part definition is that a valid individual of the whole type has the number of part individuals specified by the *cardinality* as parts with the specified *part name*. Each individual as a part must be of one of the *possible part types (allowed types for brevity)*. In other words, a part definition defines a named part “role,” which is to be filled by individual(s) of types that are viable for that role. For instance, component type (table) Lamp could have part definitions for roles Lampshade and Stand. Part definitions are either *exclusive* or *shared*. A valid exclusive part, i.e. a component individual occurring as a part with a part name that indicates it as exclusive, must not occur as a part of another component individual. In the shared case, there is no such restriction. The conceptualization also allows component types to specify whether their individuals can be shared.

The conceptualization allows *has part inheritance definitions* that specify how the properties of component individuals of whole types are dependent on the properties of parts and vice versa. These dependencies can be straightforward, such as parts inheriting the color of the whole; or they may be more complex, such as the weight of the whole being the sum of the weights of its parts.

3.4 Topology—ports

Topological concepts *port type* and a *port individual* represent how component individuals can be connected together to form a working product individual. Port definitions effectively represent the compatibility of component individuals and specify the possible topologies of the product: the idea is that component individuals can be connected only if they have compatible interfaces represented as ports. Some connections may be mandatory for creating a functional product, and other connections may be optional. Connections can be physical or logical.

Component types specify their connection possibilities by *PART DEFINITIONS*. A *PORT TYPE* is a definition of a connection interface. A *PORT INDIVIDUAL* represents a “place” wherein a component individual of some other port individual can be connected. A port type has a *COMPATIBILITY DEFINITION* that defines a set of port types whose port individuals can be connected with the port individuals of that port type.

3.5 Resources

Resource-oriented concepts model the production and use of some entity, such as power or expansion slots. The underlying idea is that some component individuals produce a *resource* and other component individuals use it. There must be enough production to cover use.

Resource production and use must be either *satisfied* or *balanced*. If the quantity of a resource produced is at least equal to the quantity of the resource used, the resource is satisfied. If the quantity of a resource produced is equal to the quantity of the resource used, the resource is balanced. A *computation definition* of a resource type specifies whether the resource must be satisfied or balanced. In addition, the computation definition specifies how the production and use of the resource type by several component individuals are combined. This is done through a *total production function* and a *total use function*, making it possible to deviate from the prototypical cumulative addition of consumption and production. Resource types can be characterized with attributes.

Component types specify with *production definitions* and *use definitions* the resource types their individuals produce and use. Generalizations in these definitions include configurable resource types to produce or consume (*a set of possible resource types*), a configurable amount (*magnitude range*), constraints on allowed attribute values of the resource to be consumed or produced (*property definitions*), and a *context*. Satisfying or balancing resource production and consumption may be restricted to a specific context; a resource is available only to component individuals that are in the same context as the producing component individual. A context can be defined through compositional structure, topological structure, type of component individuals, or a combination of these methods.

3.6 Functions

Function-oriented concepts represent the functionality that a product individual provides to the customer, the product's user, or the environment. The idea of functions is to provide a non-technical view to the functionality and features of the product to be configured. These are then mapped to component individuals, attribute values, and connections that implement the desired functionality and features. The basic concepts are *FUNCTION TYPE* and corresponding individual *FUNCTION*. Function types can specify their compositional structure with the same mechanisms as component types. However, function types cannot specify resources and ports.

3.7 Constraints

In the configuration model, *constraints* provide a general mechanism for specifying the interdependencies of entities. A constraint is a formal rule, logical or mathematical or a mixture of these, specifying a condition that must hold in a correct configuration. Constraints are used when the other concepts do not

capture the intended meaning adequately or conveniently. A constraint may specify arbitrarily complex interactions between types, individuals of types, and their properties using the terminology of the concepts.

Constraints can be divided into *constraint sets* that limit the allowed configurations from specific points of view on the product. One may check a configuration's correctness from a given point of view by checking whether the corresponding constraint set is satisfied. For example, technical and marketing constraints could form corresponding constraint sets. The technical constraints limit the configurations on the basis of which combinations are technically feasible. Marketing constraints limit the combinations on the basis of product policy, i.e., which of the technically feasible combinations a company is willing to sell.

3.8 Example

We modeled a case product, heavy rock-drilling machine Ranger by Tamrock,¹⁴ without computer support (Tiihonen et al., 1998). A Ranger (Figure 8) consists of a body, a tracked crawler base, a boom, and drilling equipment. The body is divided into a power unit, a cabin, a fuel oil tank, and a hydraulic oil tank. The Ranger product family has three alternatives in the main functional property (drilling dimension) and numerous alternatives of secondary properties. Altogether there are more than 200,000 possible variants, but Tamrock regarded 72 of these as substantially different.

Here follows an example of a part definition: Ranger has a part definition Power unit, which is fulfilled with an individual of component type Power unit assembly. The role Engine of Power unit assembly is filled with an individual of component type Engine block or one of its subtypes. Here, the abstract component type Engine block is modeled as a supertype of three concrete component types Engine R, Engine S, and Engine R w/o emission control. These types inherit the properties of Engine block but differ in some aspects. The type of the feeder is dependent on the type of Rockdrill. This dependency is modeled using port types and their compatibilities, because Rockdrill needs to be connected to the Boom. The power requirement of rock drills varies. This is modeled by resources. The engines produce different amounts of Power. The rock drills use this resource, represented at the bottom of Figure 8. The use of Power must be satisfied, i.e., it must be produced in at least the amount in which it is used.

¹⁴ Currently Sandvik Mining and Construction

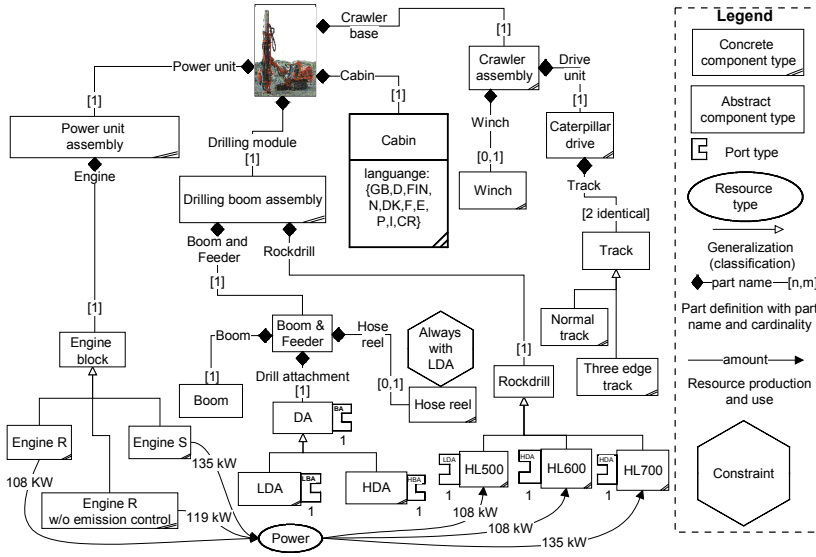


Figure 8. Tamrock Ranger modeled for demonstration and initial evaluation of the conceptualization (Tiihonen et al., 1999)

4. WeCoTin Configurator

This section presents a domain-independent sales configurator called *WeCoTin* (an acronym for Web Configuration Technology). *WeCoTin* is the main Design Science artifact of this work. A number of other Design Science artifacts are discussed; these are identified in 0 on page 19.

Requirements of *WeCoTin* are discussed in Section 4.1, followed by an overview of the system and its architecture (Section 4.2). Next, *WeCoTin* Configuration Tool for end users is described in Section 4.3. *WeCoTin* Modeling Tool for modelers and Product Configuration Modeling Language (PCML) are described in Section 4.4. User interface modeling and generation are the topic of Section 4.5. Section 4.6 outlines the management of price and delivery time. Finally, Section 4.7 briefly discusses how weight constraint rules are applied to provide the inferences required by *WeCoTin*. Note that the multi-faceted evaluation of *WeCoTin* is postponed until Section 7.1.2, and Section 8.3.2 answers research question RQ2.

4.1 Requirements

In this section, we present central requirements specific to a practical web-based configurator. The requirements have been identified in joint projects with the manufacturing industry and in the previous work of PDMG (Tiihonen & Soininen, 1996; Tiihonen & Soininen, 1997a; Tiihonen et al., 1998). We add supporting references from previous work.

Products evolve over time as new features are introduced and designs are improved or corrected. Product evolution inevitably leads to corresponding new configuration model versions—the pace of change may be high (McGuinness & Wright, 1998a). Long-term management of configuration models has often been problematic; an extreme example was the R1/XCON system (McDermott, 1993). To facilitate long-term management, product experts such as product managers should be able to model the products (Felfernig et al., 2002; Hedin, Ohlsson, & McKenna, 1998; Hvam, Riis, & Hansen, 2003; McGuinness & Wright, 1998b). This avoids the cost of experts such as knowledge engineers or programmers who are traditionally needed to maintain configurators. Modeling by product experts also eliminates the error-prone communicating of product knowledge to separate modelers.

Modeling should be easy for product experts to understand; it should also be *declarative*, allowing the modeler to specify *what* kind of product individuals are valid, instead of *procedural*, which requires specifying *how* to create them (Ardissono et al., 2003; Axling & Haridi, 1996; Felfernig et al., 2000b; Fleischanderl et al., 1998; Mailharro, 1998; McGuinness & Wright, 1998b; Sabin & Weigel, 1998; Stumptner et al., 1994; Stumptner et al., 1998; Yu & Skovgaard, 1998). The modeling language should be object-oriented to divide configuration models into relatively independent pieces with low complexity and to exploit their common properties (Hvam et al., 2008; Mailharro, 1998; McGuinness & Wright, 1998a; Slater, 1999; Stumptner et al., 1998). Furthermore, the modeling language should be straightforward to model typical configuration phenomena such as alternative components in a product structure.

The user interface for end users should require little work and no programming to create and maintain when products change. This requirement was independently documented by Attardi, Cisternino, & Simi, (1998). In addition to fluent modeling, advanced long-term management requires support for modeling the evolution of products, components, and their interdependencies in a way that resembles configuration management (CM) (Buckley, 1993) and product data management (PDM) (Männistö, 2000). It should, moreover, be possible to deploy efficiently configuration models to salespeople and customers without the risk of using outdated configuration models (Barker et al., 1989), and multiple users should be able to configure products simultaneously. Configurations should be exportable to e-commerce, enterprise resource planning (ERP), or PDM systems for further order processing (Ardissono et al., 2003; Forza & Salvador, 2006; Haag, 1998; Sabin & Weigel, 1998).

Fundamentally, a configurator must check a configuration for *completeness* (i.e., that all the necessary selections are made) and *consistency* (i.e., that no rules are violated) with respect to the configuration model (Barker et al., 1989; Fleischanderl et al., 1998; Heatley, Agarwal, & Tanniru, 1995; Sviokla, 1990). It should be impossible to order an inconsistent or incomplete configuration.

The user should be further supported by a configurator that fully deduces the consequences of previous selections (McGuinness & Wright, 1998a). This means, for example, automatically making selections implied by the previous selections, identifying alternatives incompatible with them, and ensuring at each stage of the configuration task that the user does not end up in a “dead end” that, because of previous selections, cannot be completed into a consistent configuration. In addition, explanations for any incompatibility of selections should be available (Feldkamp, Heinrich, & Meyer-Gramann, 1998; Haag, Junker, & O'Sullivan, 2006). This helps users learn the product and its restrictions. It should be possible, however, to make incompatible selections, which can help an expert user modify the configuration quickly (Forza & Salvador, 2006).

Ease and flexibility of use for non-expert users of a web-based configurator imply a number of requirements. The user should be kept aware of selections that have been made and that must still be made, and the state of completeness (complete, incomplete) and consistency (consistent, inconsistent) of the

configuration (Haag, 1998). It should be possible to guide a non-expert user through selections, but allow experts to make selections in a different order (John & Geske, 1999). Further, the configurator should be accessible to any customer who can use a web browser. Preferably, a configurator should be available in the user’s language (Hvam et al., 2008).

Finally, an interactive configurator should provide adequate performance in terms of length and predictability of response time. According to (Nielsen, 1993 p. 135), about 0.1 second is the limit that allows the user to feel that the system is reacting instantaneously, and about 1 second is the limit for the user’s flow of thought to stay uninterrupted.

4.2 WeCoTin overview

WeCoTin consists of two main components: a graphical modeling environment *Modeling Tool* (Figure 9, right) and a web-based application *WeCoTin Configuration Tool* that supports the configuration task (Figure 9, left.)

4.3 WeCoTin Configuration Tool

WeCoTin enables users to configure products over the web using a standard browser. The component *WebUIServlet* (Figure 9, upper left) acts as a presentation layer that dynamically generates the user interface for end users. The interface consists of several parts; these are indicated with a letter and description in Figure 10. The *configuration tree* (Figure 10, B) gives an overview of the configuration: compositional structure is shown, along with attributes and their values. Selections already made and selections still to be made are shown, and links facilitate a free order of making selections. The status area (top left, Figure 10, C) indicates the status of the configuration in terms of consistency and completeness and shows calculation results such as the price and estimated delivery time. The three possible states of a configuration and corresponding symbols are shown at the bottom right, Figure 10, D.

A group of *questions* related to a product individual, derived from the configuration model and user interface generation information (see Section 4.5), is represented in the *question area* (Figure 10, A). The *6-Speed Transmission* in Figure 10 is incompatible with current selections. The user is informed about inconsistencies by collecting the error messages of violated constraints—area F of Figure 10 shows an example. Configuration process in a wizard-style pre-determined order is available via the “Next” button.

4.4 Modeling Tool and PCML

Modeling Tool is used for creating and editing configuration models and information needed to generate a user interface for end users.

Configuration models are expressed in *Product Configuration Modeling Language* (PCML). PCML is object-oriented and declarative. PCML is concep-

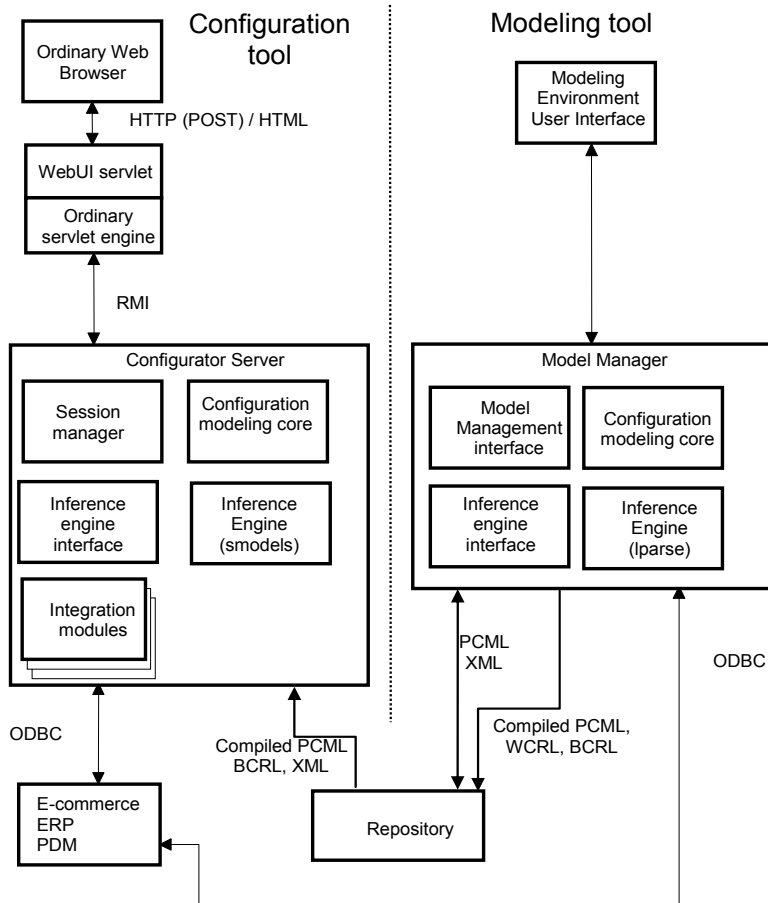
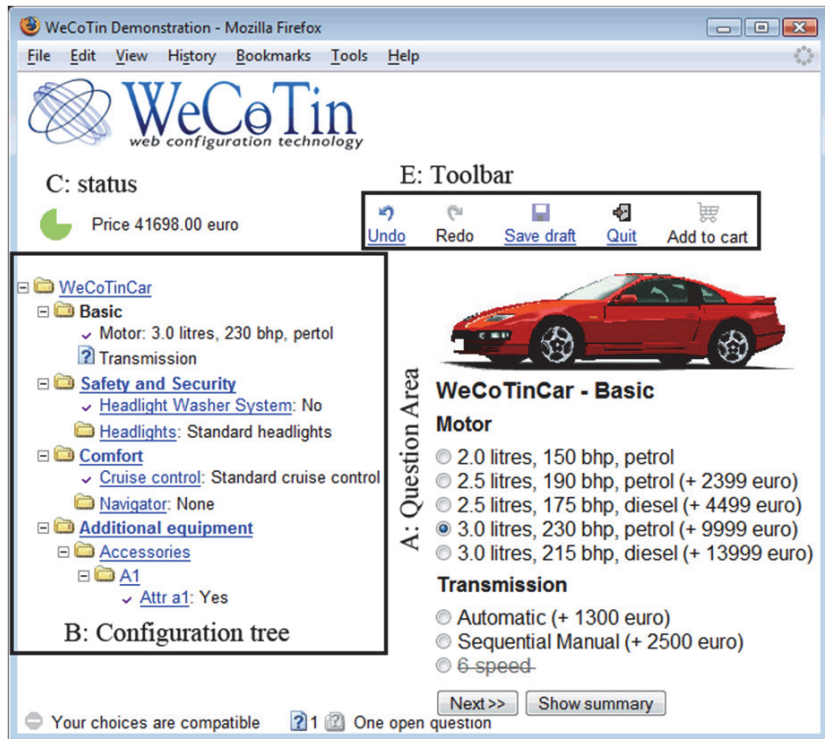


Figure 9. WeCoTin architecture overview: Configuration Tool on the left and Modeling Tool on the right. Publication II contains a single-column version of this unpublished figure.

tually based on a function-oriented subset of the configuration knowledge conceptualization (publication I, Section 3).¹⁵

Functions of the conceptualization are called *features* in the implementation. The main concepts of PCML are *feature types* and their *compositional structure*, *attributes*, and *constraints*. Feature types define the subfeatures (parts) and attributes of their *individuals* that can appear in a configuration. A feature type defines its compositional structure through a set of *subfeature definitions*. A subfeature definition specifies a *subfeature name*, a non-empty set of *possible subfeature types (allowed types for brevity)* and a *cardinality* indicating the valid number of subfeatures. One feature type is the *configuration type*: an individual directly of that type serves as the root of the compositional structure.

¹⁵ Originally, WeCoTin was developed with component-oriented modeling rather than being feature-oriented. The decision to focus WeCoTin purely on sales configuration caused the renaming of concepts. Feature types were called component types, and subfeatures were called parts.



F: error messages



D: status alternatives

**WeCoTinCar - Comfort****Cruise control**

- Active cruise control (+ 449 euro)
- Standard cruise control

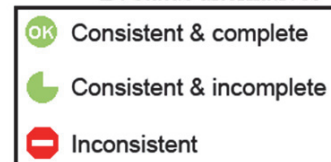


Figure 10. WeCoTin Configuration Tool user interface for end users (II). A: questions to answer and wizard-style “Next” button; B: the configuration tree gives an overview and free order of navigation; C: status, usually the price and one of the alternatives in D; E: the toolbar for other actions.

An *attribute definition* of a feature type consists of an attribute *name*, an attribute *value type*, and a *necessity definition* indicating if the attribute must be assigned a value in a complete configuration. Supported value types are Boolean, integer, and (enumerated) strings.

Feature types are organized in a *class hierarchy* in which a *subtype* inherits the attribute and subfeature definitions of its *supertypes*. A predefined *root feature type* with the name “Feature” serves as the root of the class hierarchy. A feature type is either *abstract* or *concrete*. Only an individual directly of a concrete type can be used in a configuration. Multiple inheritance among feature types is allowed. A feature type can specify defaults for attributes and subfeature realizations. *Pre-selection packages* support different sets of default

values to model market-area specific defaults and “customer standards” specifying combinations of selections that have been agreed with a customer (Pasanen, 2003). Defaults in pre-selection packages can be reinforced with soft or hard constraints (Pasanen, 2003).

PCML significantly simplifies the conceptualization of publication I. Details of aspects that have been excluded from PCML are discussed in Section 3.6 of publication II.

The *Feature type tree* displays the classification hierarchy (Figure 11, A) and serves as a starting point for editing all aspects of the types. The compositional structure is shown in the *subfeature hierarchy tree* (Figure 11, B). The feature type overview (Figure 11, C) allows addition or removal of attributes and change to the concreteness or the name of the currently selected type. The tab `Attributes` shows an overview of the attributes of the selected feature type. Special support is provided for defining enumerated attributes.

A save operation stores the graphically edited configuration model as PCML. The tool also compiles the configuration models for use in WeCoTin Configuration Tool. The semantics of PCML are provided by mapping it to Weight Constraint Rules (Soininen et al., 2001). The basic idea is to treat the sentences of the modeling language as shorthand notations for a set of sentences in the weight constraint rule language (WCRL).

Constraints associated with feature types define conditions that a correct configuration must satisfy. A constraint has a name and a constraint expression. *Hard constraints* define conditions that a correct configuration must satisfy. A configuration is considered consistent if and only if no hard constraint in any feature individual is violated. If any hard constraint is broken, the purchase process cannot be completed—for example, placing the configuration into the shopping cart is prevented. In contrast, a violated *soft constraint* issues a warning to the user, who can suppress or “silence” the warnings. However, depending on the modeler decision, the user may be required to review or even explicitly accept all warnings before proceeding in the acquisition process.

Modeling Tool provides several ways to define constraints: textually, graphically, and as table constraints. For details, see publication II.

4.5 User interface modeling and generation

A web-based user interface for the end user is generated without programming. The idea is that each selectable attribute or subfeature of a component individual being configured generates a *question*. The configurator automatically chooses a suitable input control type for each question (e.g., radio buttons, list box, etc.). On the other hand, the modeler can override the selection. Furthermore, the modeler can define for a feature type how the questions in an individual of that type are *grouped* and ordered. In the example of Figure 10, attributes `Motor` and `Transmission` of feature type `WeCoTinCar` were put to the first group named `Basic`. Thus, they are shown together in the *question area* (Figure 9, A). Normally, all questions of a group are answered before con-

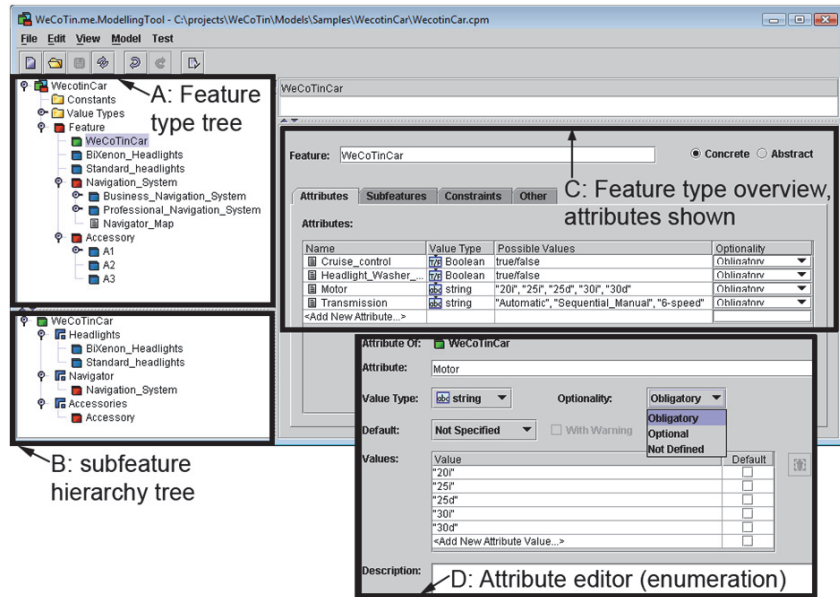


Figure 11. Configuration model in Modeling Tool showing the attributes of feature type WeCoTinCar and the enumeration attribute editor for attribute Motor (II).

sequences are deduced, but the modeler can mark some questions as requiring immediate inference. Further, the modeler can give a display name in different languages to feature types, subfeatures, possible values in attribute domains, and so forth. Display names were given, for instance, to the attribute values of `MOTOR` attribute in the example. Editing tools are provided for defining layouts and display names (aka resources). Multiple layouts and resources provide flexibility and support multiple languages for users.

Web page templates define the general look of the user interface. They support a configurator's maintainability by separating the definition of visual appearance from the product-dependent parts of the user interface.

4.6 Determining price and delivery time

WeCoTin has two mechanisms for determining prices. The *basic pricing mechanism* is applicable to simple additive prices; the more complex *advanced calculation mechanism* enables freely specifiable calculations as a function of the current configuration, the configuration model, and data in specific databases or XML files when it contacts the configuration server. Typical uses are determining the price, the delivery time, and possibly the cost. For details, please refer to publication II.

4.7 Inference with weight constraint rules and Smodels

In this subsection, we give an overview of how inference is provided for PCML configuration models.

WeCoTin Configurator Server uses as the inference engine an implementation of the weight constraint rule language (WCRL) called *Smodels* (Simons et al., 2002). *Smodels* is a system that follows the *Answer Set Programming* (ASP) paradigm, in which the problem is expressed as a theory consisting of logic program rules with clear, declarative semantics, and the stable models of the theory correspond to the solutions (*answer sets*) to the problem (Simons et al., 2002). The main functionality of the *Smodels* system is to compute for a WCRL program a desired number of stable models that are constrained by requirements specified as a *compute statement*.

The *Smodels* system is based on a two-level architecture wherein the first phase, a front-end *lparse*, compiles a WCRL program with variables into simple basic rules (BCRL) containing no variables. This potentially costly compilation process is performed off-line. The search for models of BCRL programs is handled by an efficient linear-space search procedure called *smodels*. *Smodels* is implemented in C++ and offers application program interfaces (API) through which it can be integrated into other software.

For a final step of modeling, the component *inference engine interface* in Modeling Tool (Figure 9) compiles a PCML configuration model into a WCRL program, and further, using *lparse*, to BCRL. The compilation process limits possible configurations to a finite size in a semantically justified way and applies symmetry breaking. The generated WCRL program includes the following types of sentences:

- A set of standard axioms called the ontological definitions.
- A set of sentences representing the configuration model, including the feature type hierarchy, the compositional structure, and the attributes.
- A set of sentences representing the constraints of the model.
- A set of ground facts representing the individuals out of which a configuration can be constructed.
- A set of sentences for symmetry breaking.

The BCRL form of the configuration model is loaded into the *smodels* search procedure to repetitively configure a product. The *Smodels* interface translates the user requirements, represented as attribute values and feature individuals in a configuration, into a compute statement that is sent through the API to *smodels*. The compute statement requires that some atoms be true and/or that some atoms be false. The consistency of the requirements is checked by *smodels* by trying to compute a configuration that satisfies the requirements. Deducing the consequences of requirements is based on computing an efficient approximation of the set of configurations satisfying the requirements. Intuitively, the approximation contains a set of facts that must hold for the configurations satisfying the requirements, a set of facts that cannot be true for the given requirements, and a set of unknown facts (Simons et al., 2002). Based on this approximation, the inference engine interface generates a new configuration and hands it to Configurator Server. The Configurator Server uses its calculation subsystem to compute results such as price or delivery time before the configuration is returned to the Configuration Server.

5. Recommendation of configurable offerings

Integrating recommendation functionality with configurators was motivated in Section 2.5. Next, we summarize scenarios for applying recommendation functionality to support configuration tasks (Section 5.1), and outline the applicability of recommendation technologies to the support of configuration tasks (Section 5.2). We propose extensions to the existing recommendation algorithms (Section 5.3). These extended algorithms are the Design Science artifacts of this section. This section is based on publication III. Evaluation is deferred until Section 7.1.3.

5.1 Scenarios for recommending configurable offerings

Various scenarios for the recommendation of configurable offerings were identified in publication III. A primary application of recommendation technologies is to support the user in choice navigation. In these scenarios, recommendation functionalities could focus on:

- Selecting a suitable base product to configure (such as a car model).
- Recommending a complete configuration (such as a complete PC for gaming or a tractor for peat harvesting including suitable wheels, air intake filters, and other equipment).
- Recommending how to complete a configuration (e.g., to propose still-unspecified details of a PC).
- Recommending a subconfiguration (e.g., a storage subsystem suitable for a type of use such as a PC storage subsystem for high-definition (HD) video editing).
- Recommending individual attribute values or component selections (e.g., a mobile data connection for a salesperson).

An obvious way to convey recommendations is to present them as defaults in the user interface. Numerous other alternatives exist, such as ordering of the alternatives so that recommended ones are shown first.

Another type of scenario for applying recommendation technologies in the context of configuration systems is related to inconsistent requirements. If no configuration can be found for a given set of user requirements, a possible approach is to calculate and present diagnoses (repair alternatives) (Felfernig et

al., 2009; Felfernig & Schubert, 2011; Tiihonen, Felfernig, & Mandl, 2014). A *diagnosis* is a set of user requirements that have to be removed or changed to enable a configuration to be found. Of special interest are minimal diagnoses that contain as few alterations as possible. Quite often a large number of alternative diagnoses are possible. Personalized recommendation and personalization can help identify preferred diagnoses.

Yet another scenario is to apply recommendation technologies to select (rank) non-configurable products that become part of a configuration (IV). For example, in the configuration of a mobile package, a phone (or a mobile terminal) is selected. Alternatives are ranked in a personalized manner, such as by the multi-attribute utility theory (MAUT) or by similarity of product features to the user's preferences.

Finally, numerous authors have proposed to exclude (filter out) some features (e.g., attributes) or alternatives (e.g., attribute values) that can be determined uninteresting from the customer's point of view (see, e.g., Resnick & Varian (1997), Stegmann et al. (2006), Falkner et al. (2011)). A similar idea is to recommend features (configuration decisions)—what remaining selectable attribute or part selection would be of most interest to the user to configure next (Falkner et al., 2011; Felfernig & Burke, 2008; Wang & Tseng, 2011).

5.2 Selection of recommendation techniques

Characteristics of the main recommendation technologies (see Section 2.5.1) were analyzed with respect to the requirements of supporting interactive sales configuration (III). Here, we concentrate on techniques that can be applied to support feature value selection.

The widely applied *collaborative filtering* and *content-based recommendation* techniques are challenging to apply in the context of many sales configuration scenarios: Traditional collaborative filtering becomes challenging if an individual customer purchases too few configurable products to establish a dense enough user profile to be suitable for generating recommendations. A major challenge of applying content-based filtering to recommend configurable offerings is that—besides the availability of accurate textual component descriptions—building a user profile requires repetitive configurations of one user. Furthermore, a profile may soon become outdated in rapidly evolving domains such as PCs.

Utility-based recommendation technologies (see Section 2.5.1) can be applied in the context of recommendation, for example to suggest individual attribute values or component selections based on their utilities. Felfernig et al. (2007) give an example in the financial-services domain. In addition, Ardissono et al. (2003) discuss utility-based approaches in the configuration context.

Knowledge-based recommenders exploit explicit information about items and user requirements and how those can be satisfied (Burke, 2000; Felfernig et al., 2007). Creating and maintaining explicitly formulated recommendation

knowledge may be resource intensive. On the other hand, users may benefit from expert knowledge embedded in recommendations.

In publication III, we examined the idea of case-based recommendation of configurable offerings because the technique does not require explicit knowledge engineering. Furthermore, the case-based techniques adapt the idea of collaborative recommendation: selections of other, similar users are applied to determine recommendations on the level of individual features and feature value combinations.

5.3 Proposed extensions to recommendation algorithms

We propose extensions to the case-based recommendation algorithms of Cöster et al. (2002). The goal was to indicate potential improvements for improving prediction quality in future system developments. Next, we outline the extensions (III).

We considered, for simplicity, only “flat” configuration models consisting of a fixed set of features (static structure and connections), each having a finite domain of possible values.

When determining recommendations, the algorithms of Cöster et al. (2002) take into account equal feature values, and all features are considered equally important. The proposed extensions include (1) *feature importance weights* to take into account the varying importance of features to the customer, and (2) taking into account *similarity of feature values* (i.e. *offered alternatives*). Furthermore, a simple nearest-neighbor algorithm was introduced.

The intuition of taking into account feature importance weights is that, because the importance of configurable aspects from a customer’s point of view varies, determining recommendations should take this into account. Aspects with higher importance (which is represented as greater weight) should influence the recommendations more significantly than those with less weight.

The intuition of taking similarity into account is that, for example, in the case of recommending a hard disk, a 500 gigabyte (GB) hard disk may be quite similar to a 400 GB or 600 GB hard disk, but 200 GB or 1,500 GB would probably satisfy different purposes.

Next, an example product and applied notation will be presented (Section 5.3.1), and the concepts of similarity and distance are introduced (Section 5.3.2). Then we outline how similarity was integrated into *weighted majority voter* (Section 5.3.4) and *naïve Bayes voter* (Section 5.3.5) algorithms, and importance weights into the *nearest neighbor* (Section 5.3.3) and *weighted majority voter* algorithms. For brevity, we omit discussion of *most popular* choice, which is similar to the naïve Bayes voter.

5.3.1 Example product and notation

The product of the example is a PC that has as selectable features

- a motherboard out of 4 types ($mb \in \{a1, a2, i1, i2\}$),
- a hard disk out of 3 types ($hd \in \{h2, h5, h9\}$),

- an optical drive out of 4 types ($od \in \{dr, dw, br, bw\}$),
- a processor out of 3 types ($pr \in \{as, i4, i9\}$), and
- Optionally, a graphics card out of 3 types ($gc \in \{none, g2, g8, g9\}$).
- The amount of memory is specified in gigabytes ($me \in \{1, 2, 3, 4\}$).

Intended usage is characterized by three features:

- video editing: no, standard definition, or high definition ($vi \in \{no, sd, hd\}$);
- photo editing: no, standard, or advanced ($ph \in \{no, std, adv\}$); and
- gaming (no, or 2d), 3d, or advanced 3d ($ga \in \{2d, 3d, adv\}$).

A complete configuration specifies a value for each feature. Furthermore, a *valid* configuration is complete and consistent with a defined set of constraints. For brevity, we omit the constraints and further properties of the components because they are not necessary for understanding the recommendation algorithms and the proposed extensions. For the interested reader, constraints and other details of the product of the example are available in publication III. Table 4 exhibits five previous valid configurations, and an incomplete configuration of the active user u , for whom suitable feature values will be recommended.

Previous complete and consistent configurations specify consistent values for all the existing N features f_1, \dots, f_N (see Table 4). The k th configuration is referred to as $conf_k$. The value of feature f_i in configuration k is referred to as $f_{i,k}$, and index i can also be referred to with the name of the feature. For example, f_1 represents feature video (vi), and thus $f_1 = f_{vi}$ and $f_{1,3} = f_{vi,3} = sd$ (see Table 4). When referring to the profile of the active user, we use index u : for example, $f_{i,u}$ refers to the value of feature f_i for the active user. The set of specified features in the active user profile is F_u , in the example $\{f_{vi}, f_{ph}, f_{ga}\}$; see the last row of Table 4. The set of features for which the active user profile does not have a value is \bar{F}_u , in the example $\{f_{pr}, f_{mb}, f_{me}, f_{hd}, f_{gc}, f_{od}\}$. Finally, $dom(f_j)$ returns the domain (possible values) of feature f_j .

The *nearest neighbor* and *weighted majority voter* algorithms apply *feature importance weights*. The example applies the following distribution: video editing $w(f_{vi}) = 5\%$, photos $w(f_{ph}) = 5\%$, gaming $w(f_{ga}) = 9\%$, processor $w(f_{pr}) = 18\%$, motherboard $w(f_{mb}) = 5\%$, amount of memory $w(f_{me}) = 15\%$, hard disk $w(f_{hd}) = 16\%$, graphics card $w(f_{gc}) = 17\%$, and optical drive $w(f_{od}) = 10\%$. These weights could stem from direct customer specifications, representative preferences from statistical samples, or the application of utility constraints, as documented by Felfernig et al. (2007).

Table 4. Configurations from the previous configuration sessions, and the active-user profile (*Conf*). Adapted from publication III.

<i>k</i>	<i>f</i> ₁ <i>vi</i>	<i>f</i> ₂ <i>ph</i>	<i>f</i> ₃ <i>ga</i>	<i>f</i> ₄ <i>pr</i>	<i>f</i> ₅ <i>mb</i>	<i>f</i> ₆ <i>me</i>	<i>f</i> ₇ <i>hd</i>	<i>f</i> ₈ <i>gc</i>	<i>f</i> ₉ <i>od</i>	<i>cl</i>
1	no	no	2d	as	a1	1	h2	none	dr	ba
2	no	std	2d	as	a2	1	h5	g2	dw	st
3	sd	std	adv	i4	i2	3	h5	g9	dw	ad
4	hd	adv	adv	i9	i2	4	h9	g9	bw	ad
5	sd	adv	3d	i4	i1	2	h9	g8	dw	st
u	no	no	3d							

5.3.2 Distance and similarity

To support the idea of taking into account similarity of feature values in determining recommendations, we aim to express similarity numerically. We determine similarity through the concept of *distance*. Distance $d_{f_i}(x, y)$ between any two feature values x and y of feature f_i is *normalized* to be usually in range 0 to 1.¹⁶ Here, a distance of 0 means equal values, and the distance between (for instance) the smallest and the largest value (maximum distance) would be approximately 1. We define *similarity* as $sim_{f_i}(x, y) = 1 - d_{f_i}(x, y)$. Thus, equal values have a similarity of 1, and the most distant values have a similarity of about 0.

We determine distance $d_{f_i}(x, y)$ with the *heterogeneous value difference metric* (HVDM) (Wilson & Martinez, 1997). HVDM copes with symbolic (nominal) and numeric features in a relatively simple yet uniform manner. To determine distance between symbolic values, the HVDM needs training: distance is determined based on the correlation of attribute values and their *classification*. To oversimplify slightly, the closer the probability of a pair of feature values being present in identically classified configurations, the more similar these feature values are considered to be. In the example, we take a simplistic view, and consider a configuration to belong to one of three classifications—basic (*ba*), standard (*st*), or advanced (*ad*)—that represent the configuration’s sophistication level. This classification is used as the classifier for HVDM (see column *cl* in Table 4.)

5.3.3 Nearest Neighbor

The idea of the *nearest neighbor* is simple: determine a neighbor configuration, which is closest to the known parts of the active user’s profile, and recommend the feature values of this nearest neighbor for the remaining features. The distance of the configuration $conf_u$ of the active user and neighbor configuration $conf_a$ is defined as the sum of distances between corresponding feature values, weighted by feature importance weights ($w(f_i)$). Only features that have a value in the active user’s configuration are taken into account.

¹⁶ According to (Wilson & Martinez, 1997), it is customary to normalize values so that possible outliers (exceptionally large or small values) do not have an effect on the range. For example, 95% of values are used to determine the range.

A nearest-neighbor configuration with the smallest distance is identified among those configurations which complete the active user's configuration in a consistent manner. Recommendations are feature values of this nearest-neighbor configuration. In the example, the nearest neighbor relative to the user profile is $conf_1$: $d_{vi}(no, no) = 0.000$, $w(f_{vi}) = 0.050$; $d_{ph}(no, no) = 0.000$, $w(f_{ph}) = 0.050$; $d_{ga}(3d, 2d) = 0.707$, $w(f_{ga}) = 0.090$.¹⁷ Total weighted distance $dist(conf_u, conf_1) = 0.064$. But completing the user configuration with the values of $conf_1$ would create an inconsistent configuration (for details, see III). Therefore the feature values of the nearest consistent neighbor $conf_5$ are recommended, e.g. processor i_4 and motherboard i_1 .

5.3.4 Weighted Majority Voter

The *weighted majority voter* (Cöster et al., 2002) recommends individual feature values based on each neighbor configuration "voting" for its feature values. One vote to give (weight 1) is gained for each feature value that is identical in the voting configuration and the active user's already-specified configuration. For example, if a configuration has 4 features that have the same value as already specified by the active user, the configuration votes with 4 votes for its feature values. A feature value with most votes from all neighbors is recommended. For example, the weight of $conf_1$ for the active user u is 2, because photo editing and video editing match ($f_{vi,1} = f_{vi,u} = no$, and $f_{ph,1} = f_{ph,u} = no$). Thus, $conf_1$ would contribute by recommending its feature values by giving 2 votes to its feature value settings of the yet-unspecified features: $f_{pr} = as$, $f_{mb} = a1$, $f_{me} = 1$, $f_{hd} = h2$, $f_{gc} = none$, and $f_{od} = dr$. As a sum of all neighbor configurations voting for their feature values with their neighbor weight, the following feature values get most votes: $f_{pr} = as$ (3 votes), $f_{mb} = a1$ (2), $f_{me} = 1$ (3), $f_{hd} = h2$ (2), $f_{gc} = none$ (2), and $f_{od} = dr$ (2).

In publication III, we proposed an extension to the approach of Cöster et al. (2002). The extension is an alternative way of determining the neighbor weights: *Neighbor weights (votes) are determined by the similarity of neighbor and user profile feature values instead of equality*. This could improve prediction quality in the presence of similar feature values. Values similar to the user's existing selections in a previous configuration would also contribute to the weight of the neighbor. Second, the importance of individual features for a user (feature importance weights) is taken into account. Here, the similarity of feature values is further weighted (multiplied) with the feature importance weight.

For example, the weight neighbor of $conf_1$ ($w(conf_1, conf_u)$) = 0.126. With the alternative weights, processor as ($f_{pr=as}$) gets most votes (0.191). Here, the modified algorithm provided the same feature values with most votes as the original algorithm.

¹⁷ For brevity, the calculation of distance d_{f_i} between individual feature values is omitted; see publication III.

5.3.5 Naïve Bayes voter

The naïve Bayes voter (Cöster et al., 2002) recommends individual feature values. To determine a recommendation for feature f_j , a *probability predictor* is determined for each possible feature value v ($v \in \text{dom}(f_i)$). A feature value with the highest probability predictor will be recommended.

The naïve Bayes voter applies the idea of the Bayes theorem:

$$P(B | A) = \frac{P(A | B) P(B)}{P(A)}$$

Here, A and B are Boolean-valued random variables representing occurrence of corresponding events, and $P(A)$ and $P(B)$ are the probabilities of these events. $P(B|A)$ denotes the conditional probability of event B given that event A has taken place, and $P(A|B)$ the conditional probability of event A given event B .

In the case of the naïve Bayes voter, event B represents the fact that feature f_j has value v . Event A represents the fact that a configuration has the combination of feature values already specified by the active user in F_u . Thus, $P(A|B)$ is the conditional probability of the active user's current value combination for already specified features, given that feature f_j has value v . Finally, $P(B|A)$ is the probability of the feature f_j having value v , given the partial configuration of the active user.

Applying this idea, the naïve Bayes voter calculates a basic probability $P(B)$ and a conditional probability $P(A|B)$. The divisor $P(A)$ is omitted to simplify calculations—it would be the same for all feature values to be compared.

The conditional probability part $P(A|B)$ determines a probability of the active user's current value combination (the feature values of F_u), given those neighbor configurations that have feature $f_j = v$. For each feature $f_i \in F_u$, a conditional probability estimate $P(f_{i,u} | f_j = v)$ for the current value of the active user ($f_{i,u}$) is calculated. The conditional probability for the combination of values in F_u is the product of individual feature value probability estimates, utilizing the naïve Bayes independence assumption.¹⁸ This part of the original formula is not affected by the proposed extension. For details, please see publication III.

In the original formula (Cöster et al., 2002), the basic probability $P(B)$ for value v of feature f_j is simply the proportion of configurations having that feature value. For example, feature optical drive f_{od} for value $v = dw$ gives $Pr_{basic}(f_{od}, dw) = 0.600$ because 3 of 5 neighbor configurations have value dw for f_{od} .

The proposed extension of the naïve Bayes voter (III) potentially improves the prediction accuracy in the presence of similar feature values. Here, we take into account similar feature values in neighbor configurations instead of requiring them to be equal when determining feature values to recommend. Note that adding feature importance weights remains future work.

¹⁸ The Naïve Bayes independence assumption considers that features are conditionally independent of each other, i.e. value of one feature does not affect the distribution of values of other features. While this assumption is usually false, in many applications good results can be obtained.

To take similar feature values into account, we modify the determination of the basic probability $P(B)$. We give each feature value *support* when neighbor configurations have feature values within a predefined maximum distance Δ , instead of requiring equal feature values. The idea is to diminish support quickly when the distance increases, and hence a quadratic formula was proposed: $s_{f_j}(x, y) = (1 - d_{f_j}(x, y))^2$, if $d_{f_j}(x, y) \leq \Delta$. When $\Delta = 0$, only equal values are taken into account. Supports of different feature values are scaled so that the sum of scaled supports is 1. Applying the similarity-based basic probability formula with a (very large) $\Delta = 0.8$ yields alternative basic probabilities, such as $Pr_{basic}(f_{od}, dw) = 0.578$. In this example, the same recommendations result as with the original formula. For example, optical drive dw is recommended.

6. Sales configuration of services

This section discusses configuration of services with the focus on the following questions:

- What variation is offered for services in practice?
- Can configurators designed for physical products be applied to manage the offered variation of services?

Publication V analyzed three real service cases with configurable offerings to identify the variation offered and the configuration-related processes.¹⁹

The mode of research was exploratory. The first case concerned the maintenance services of an elevator manufacturer in a business-to-business setting. The second case involved insurance, and the third case telecommunications services, both representing business-to-consumer (B2C) offerings.

In all these cases, *service contracts were configured*, and they exhibited characteristics of configurable products as defined by Tiihonen et al. (1998):

- Each contract was adapted to the individual needs of a customer.
- The offering was pre-designed to meet a given range of different customer requirements.
- Each solution was specified as a combination of pre-designed *service elements* (corresponding to component types and individuals, more or less distinct building blocks of the service solution or another service element) and their parameters.
- There was no need to design new service elements within the scope of the sales delivery process.

6.1 Offered variation in configurable service products

The classical Ws (what, when, who, where, how, and why) have been found useful characteristics for describing services (Dumas et al., 2002). We followed this idea to characterize variation offered in the service contracts of the case companies. Each W was considered as a potential source of offered variation.

¹⁹ An extended and improved version of publication V was published at the MCPC 2007 conference (Tiihonen et al., 2007). The review process of MCPC 2007 provided no reviewer comments, only acceptance. Therefore, we present publication V as part of this thesis. In comparison with publication V, the MCPC version adds details about taking configurators designed for physical products and applying them to services.

Next, in the context of each W, the line of thought showing the potential for offered variation is identified, and examples from the cases are presented. The views are not mutually exclusive: a service element or a parameter can pertain to several views.

What-variation

What the customer receives as the outcome of the service is a central aspect of the “service package” (Grönroos, 2007). *What-variation* corresponds to traditional configuration of physical products—defining what the customer gets. Typically, *what-variation* manifested itself via optional and alternative service elements or as parameter values of service elements. For example, assisting workforce for official inspections arranged by the service provider was an optional service element in the context of maintenance contracts. The broadband connections had available three alternative security service elements.

When-variation

Time plays a central role in most services (Lovelock & Gummesson, 2004). *When-variation* relates to the temporal aspects of service as a whole or some of its elements, such as temporal availability or response time. For example, the maintenance case included as options corrective maintenance during evenings or weekends.

With what? Who? How?

Grönroos’ definition of service (refer to Section 2.6.1 on page 34) includes that service employees, physical resources or goods and/or systems of the service provider interact with the customer. It is possible to offer variation on both human and physical resources (who, with what) and the way of interacting (how), for example, in terms of such characteristics as the skills or qualifications of human resources or in terms of the characteristics of physical elements such as the type, quality or sophistication of equipment used in service delivery, or the way of delivering service elements.

With-what and how-variation were offered in the context of reporting and payment. For example, it was possible to specify with what and how stakeholders were informed about major maintenance events, such as an e-mail and/or a text message (SMS) after a repair’s completion. Billing could be configured to be electronic or paper-based, and payment to be regular or direct-debit.

The cases, initially surprisingly, did not have offered variation on the characteristics of human resources. A natural explanation is that that core service was based on human delivery in the elevator maintenance case only. In the case, customers buy from a large company and it may be natural to assume that the personnel is qualified and has sufficient experience. Thus, the nature of the case services may explain that there was no offered variation of human resources.

To Whom

As Chowdhury & Miles (2006) summarized previous work, “customer-induced uncertainty depends on the diversity of customers, whether customers are opportunistic, and the significance of each customer” (p. 123). We noticed that the properties of the service recipient, whether a person, equipment, or information, could significantly affect the availability of the service, service delivery, or pricing. Furthermore, a number of stakeholders beyond the direct service recipient might have an effect. For example, they might affect service requirements, reporting, or purchase decisions.

The service recipient was always specified in the service contracts. The availability of some service elements depended on the properties of the service recipient. For example, all-inclusive maintenance contracts were not available for old elevators, and voluntary health insurance was not available to persons above a specific age. In insurance and maintenance cases, pricing was affected by the properties of the service recipient.

Where

A significant factor of service delivery process concerns where service is delivered, especially in terms of *who goes to whom to facilitate service delivery* (Lovelock, 1983). Service delivery location may have a significant effect on total customer sacrifice and is thus a potential source of offered variation.

“Where” or “who moves” variation was not actively offered in the cases. The maintenance case concerned large, permanently installed equipment that must be maintained at the site. Similarly, this aspect was not central in the insurance and broadband services. Some car-related insurance policies allowed the insurance company to determine where a repair would be performed, while other policies allowed the customer to make the decision.

Why

We did not encounter any explicit *why*-view-related sources of offered variation in the configurable service offerings. It was evident, however, that reasons for buying a service solution affect the suitable solution.

6.2 Sources of variation in relationship-based services

We identified a number of potential and realized sources of offered variation in the context of services based on a formal relationship. These include pricing, paying and billing, information and reporting, ownership and intellectual property rights, and loyal customer benefits.

Paying and invoicing

Paying and invoicing were *hidden services* (Grönroos, 2007) in which variation was offered. The insurance case (B2C) offered a configurable number of yearly payments and a selectable due date. The telecommunications case (B2C) offered alternative levels of billing itemization, e.g., with respect to per-use charges. The maintenance case offered paper-based or electronic invoices and options for grouping of billing targets for business-to-business customers.

Information and reporting

Provision of information and extranet services are both often hidden services (Grönroos, 2007). In the maintenance case, configurable notifications about service events were available, e.g., for equipment breakdowns. The scope of information and push-style reports available via extranet were configurable, and alarms on repair costs exceeding a pre-determined value or a specific number of faults were available.

Pricing and discounts

Pricing models and discounts for services and products are a complex phenomenon (de Miranda, Baida, & Gordijn, 2006; Dumas et al., 2002; Lovelock, 1983). We encountered three basic types of price elements: one-time, recurring (periodic), and pay-by-use. The telecommunications case included one-time as *initiation price elements* when the service contract was initiated. *Periodic price elements* such as monthly, quarterly, or yearly fees were common. Transaction-based *pay-per-use price elements* were also common. For example, in many pricing schemes, mobile phone calls and text messages (SMS) were charged by use. Allocation of the total service cost to different kinds of price elements varied significantly. Often different combinations of periodic and pay-per-use were offered: increased periodic payments included increased amount of included use or offered reduced pay-per-use rates. In addition, the maintenance case offered different combinations of transaction and periodic fees.

Loyal customer benefits

Various loyal customer benefits can be offered, thanks to the long-term nature of contract-based services. In the B2C telecommunications case, the company offered a number of mutually exclusive loyal-customer benefit programs.

Ownership and intellectual property rights

Ownership and intellectual property rights of information or intangible deliverables can be agreed. For example, who owns databases gathered in remote monitoring of equipment or the detailed maintenance history? These were not configurable options in the cases. In a case company, these considerations required case-specific negotiations.

6.3 Services and configurators for physical products

We experimented with modeling of elevator maintenance contracts, mobile subscriptions, and insurance policies with the WeCoTin configurator. The goal was to evaluate if existing real-world offerings could be modeled and configured, as well as to identify possible shortcomings. WeCoTin served as an example of a modern configurator designed for configuring goods.

Modeling of contract-based service offerings was possible, and no significant challenges were encountered. It was straightforward to model service elements as feature types, some of which had parameters (modeled as configurable attributes). Optional and alternative service elements were modeled as allowed

types in subfeature definitions with cardinality “0 to 1” or “1 to 1.” No cardinalities were encountered with a maximum higher than 1. The compositional structure was narrow and shallow.

Parameters were always needed for the service solution represented by the configuration type. Often some service elements were parametric. Thus, attributes were a very useful modeling mechanism. Classification hierarchy and refinement were useful; we often modeled different service products of a family as subtypes of a common supertype. Refinement reflected different variation possibilities in subtypes; the possible values of attributes (domains), and sometimes the allowed types of subfeature definitions were refined.

Configuration rules required for ensuring consistent specifications were not common: there was little need for constraints to enforce them. However, the customer or other stakeholders, related equipment, environment, or their properties often had to be modeled to verify that some services, service elements, or values for their parameters were available, or that they could be priced. This resulted in feature types that actually represented properties of stakeholders. Constraints were used to model the dependencies.

We did not model the prices of the offerings to constrain the required modeling effort. As identified in Section 6.2, instead of one price typical for goods configuration, the telecommunications case would have required at least two: the initiation and periodical fees needed to be kept separate. Estimating the total cost of customership of different solutions would have been useful.

Another modeling requirement new to us was the need to assign different stakeholders as the resources of service activities. Roles of stakeholders can vary through, say, different selections in *what-* or *by-whom*-variation. The same stakeholder can act as a resource in several activities. Therefore modeling with compositional structure with exclusive parthood of PCML (one individual cannot be a part of several whole individuals simultaneously) is not practical.

Based on hands-on modeling and vendor claims, we conclude that at least some (and probably most) configurable service offerings can be modeled and configured with traditional configurators. We felt a conceptual mismatch in modeling, however, because thinking in feature types (or component types) did not seem natural for services. Furthermore, the required scope of modeling is broader, because relevant stakeholders, equipment, or environment must be modeled to verify availability or determine pricing.

The conceptual mismatch led to the development of *four-worlds model* (4WM) for sales configuration of services (Heiskala, 2005; Heiskala et al., 2005; Heiskala et al., 2006). 4WM divides object types into four mutually exclusive *worlds*. The *service solutions world* contains “normal” configuration modeling concepts to capture the specifications to which the service is to be delivered. Extensions include concepts for describing the recipient(s) of service (such as persons or physical systems) and the environment relevant to the recipient(s). These are captured by the *objects-of-service world*. Requirements to be satisfied, such as goals or benefits sought from the service, non-functional requirements such as performance, security or reliability, prefer-

ences, and other factors captured by the *needs world*. Further, the *process world* makes it possible to configure the service delivery process and the resources required to carry it out. 4WM seems to provide an adequate conceptual basis for modeling configurable services for sales purposes.

Based on the concepts of 4WM, the *Service Configuration Modeling Language SCML* (unit ID 13 in Figure 6, p. 22) was specified (Anderson, 2005). It makes it possible to express configuration models of services with a PCML-like syntax and improved conceptual match. A front-end for WeCoTin was developed that compiled configuration models expressed in SCML into PCML (Anderson, 2005). This enabled configuration of services with WeCoTin based on configuration models expressed with SCML. To constrain the scope of this work, a more detailed discussion about 4WM and related artifacts is omitted.

7. Evaluation

This work applies two proposed points of view of Design Science research evaluation. In addition, the relation of this work to the guidelines of Design Science research of Hevner et al. (2004) will be outlined. The discussion embeds the identification of the contributions.

Hevner et al. (2004) emphasize the artifact as the output of research and see *utility* as the main criterion. They also recognize additions to the “knowledge base” as contributions. They provide seven guidelines of Design Science research. According to the guideline “#3 Design Evaluation,” the utility, quality, and efficacy of a design artifact must be rigorously demonstrated via well-executed evaluation methods. Peffers, Tuunanen, Rothenberger, & Chatterjee, (2007) build on Hevner et al. by considering that a complete Design Science research methodology requires three major components: principles, practices, and procedures. Of these, “a procedure that provides a generally accepted process for carrying it²⁰ out” was missing from Hevner et al. (p. 50). Peffers et al. (2007) see evaluation as “observ[ing] and measur[ing] how well the artifact supports a solution to the problem. This activity involves comparing the objectives of a solution to actual observed results from the use of the artifact in the demonstration” (p. 56). The guideline of Hevner et al. and the framework of Peffers et al. form a basis to evaluate the artifacts of this work.

On a higher level of abstraction, Gregor (2006) identifies five theory types in research. Relevant to Design Science is *theory type V: design and action*, which “Says how to do something. The theory gives explicit prescriptions (e.g., methods, techniques, principles of form and function) for constructing an artifact.” (p. 620). Gregor and Jones (2007, p. 322) further elaborate that the primary output of Design Science is *Information Systems Design Theory* (ISDT). ISDT “shows the principles inherent in the design of an IS artifact that accomplishes some end, based on knowledge of both IT and human behavior. The ISDT allows the prescription of guidelines for further artifacts of the same type.” Thus, contributions are not the artifacts themselves. Rather, contributions are more general prescriptions for artifacts of the same type.

Artifact evaluation takes place in Section 7.1. An ISDT for sales configurators is presented in Section 7.2, and the relation to the guidelines of Design Science research is outlined in Section 7.3.

²⁰ 'it' refers to 'research'

7.1 Artifact evaluation

The next sub-subsections evaluate the main artifacts with respect to the third guideline of Design Science research (Hevner et al., 2004) and the methodology of Peffers et al. (2007). Finally the contributions of this work's service configuration branch are identified.

7.1.1 Conceptualization for configuration knowledge

The utility of a configuration conceptualization is not easy to characterize: benefits, if any, will be realized only when the conceptualization is implemented as a component of deployed systems. Stvilia (2007) suggests that it is essential for an ontology evaluation model that the model measures “how completely, consistently, or accurately the ontology represents the domain concepts in relation to the general cultural context and the context of a particular activity system”. In this spirit, real cases were modeled.

Demonstration and initial evaluation of the conceptualization was provided by modeling a case product (heavy rock-drilling machine Ranger by Tamrock) without computer support; see the example in Section 3.8 (p. 42). The conceptualization was found to cover the relevant modeling needs fairly well (Tiihonen et al., 1998). It was possible to model the offered variation of the case product with the conceptualization. The conceptualization matched the modeling needs of the case product. Compositional structure was used significantly. The need for ports or resources was not as crucial—even some effort was required to find realistic examples. This absence of need for ports and resources may be explained by a structure-oriented thinking pattern in the case company. Information technology support for modeling was considered necessary for the full-scale use of the conceptualization.

Two further modeling efforts were performed independently outside PDMG (Niemi, 2007). The modeled products were reach trucks for multi-purpose material handling, the Rocla Humanic product family, and a Nokia mobile-phone hardware product platform. It was possible to model the offered variation of the products with the conceptualization. Modeling was performed without IT support, and it was considered that such support would be essential for “every-day-usage.” Ports were very useful in the mobile-phone case. Resources were not applied. Challenges for modeling were encountered in expressing allowed module combinations, because numerous constraints were needed.

All modeling efforts indicated that the conceptualization could capture the products' offered variation completely and accurately. The relevance and usefulness of the central concepts varied by domain, as could be expected. But fluent modeling would require appropriate computerized support for defining the individuals of concepts, their compositional and taxonomical hierarchies, and the constraints. The author of this thesis considers the conceptualization to be a major contribution to the state of the art during the publication of publication I.

7.1.2 WeCoTin Configurator

Evaluation of WeCoTin was multi-faceted. High-level efficacy and utility of WeCoTin were demonstrated by practical case examples. Next, more details will be provided on model characterization and the usefulness of modeling mechanisms, evaluation of Modeling Tool, and empirical evaluation of run-time performance.

Model characterization and usefulness of modeling mechanisms

WeCoTin and PCML were used to model and support the configuration tasks of several industrial domains (publication II, Tiihonen, 2009; Tiihonen, 2010). The sales configuration view of 14 real-world products was modeled in their entirety (some with extra demonstration features, one in 2 variants), and 8 partial products or concepts. These offerings came from 10 companies or organizations representing machine industry, healthcare, telecommunications services, insurance services, maintenance services, software configuration, and construction. One configuration model was exceptionally large and semi-automatically generated, identified as the *Linux* model.

A number of characterizing metrics based on static configuration model analysis were developed. Modeling Tool was instrumented to calculate the metrics. A subset of the metrics is presented in Table 5.

Average numbers below do not include the exceptionally large Linux model. The configuration models were typically relatively small. The *numbers of abstract, concrete, and total feature types* (columns “Abstract types,” “Concrete types,” “Total types,” respectively, in Table 5) contribute to the size of a configuration model. Another way to characterize the size of configuration models is through the number of *questions*. For example, a selectable attribute value or a subfeature of a feature individual being configured generates a question during a configuration process. The average *number of questions in a configuration model* (‘Questions’ in Table 5) was 61 questions per configuration model and 5.4 questions per concrete type. Often both attribute and subfeature definitions concentrated on the configuration type (“% root questions” in Table 5).

Attributes were the main mechanism for modeling offered variation; all 26 models defined attributes, and on the average 83% of questions originated from attributes and the remaining 17% from subfeatures (parts). The number of *effective attributes* of the configuration model (“Effective attributes” in Table 5) is the sum of inherited and locally defined attributes in concrete types.

Although applied more sparingly than attributes, compositional structure was an important modeling mechanism. An indication of the use of compositional structure is given by the number of effective (inherited and locally defined) subfeatures in concrete types (“Effective subfeatures” in Table 5.)

Taxonomical hierarchy and inheritance were used significantly. On the average, 59% of feature types were *subtypes of some abstract type* (other than the root of the feature type hierarchy *Feature*, “Subtypes,” and “% as subtypes” in Table 5.) Inheritance significantly saved modeling effort necessary for larger models. In these models, almost half (49%) of effective attributes were inherited (“% inherited” in Table 5.) Refinement was useful for limiting the domain of

allowed values or allowed types in subtypes. Inheritance of the compositional structure (“% inherited subfeatures” in Table 5) was also useful, although it was applied only in about 31% of the models because the compositional structure was shallow and often concentrated on the configuration type.

The number of constraints (“Constraints” in Table 5) varied widely (median was 13 per model), and inheritance of constraints was applied to some extent; abstract feature types defined about 2 constraints per model.

Floating or fixed-point numbers or integers with very large domains would have been useful in the insurance and compressor domains. The cases included, for instance, a freely specifiable amount of insurance coverage, specification of the apartment size in m², or the calculated capacity of a compressor. These were managed by discretizing. Neither explicit resource modeling nor topological modeling (e.g., ports) was needed.

The “basic” price calculation mechanism with only additive prices was applied for four real products and three demonstration models (“bas” in the column “Price” of Table 5). The “advanced” calculation mechanism (Nurmilaakso, 2004) determined the price for three products (“adv” in the column “Price”). Prices were often omitted, either in response to indicated sensitivity or in order to constrain modeler resource usage. The basic price calculation mechanism would have been sufficient for other products except compressors and insurance products.

Evaluation of Modeling Tool

The researchers using Modeling Tool considered the editing facilities generally very adequate. The visual-type and compositional-structure hierarchies, the fluent attribute domain editing, and the graphical constraint editor all facilitated efficient modeling. Table constraints were very useful. Global renaming support of objects without the need for text-based search and replace was convenient and likely to reduce potential for errors.

User interface development was rapid because product dependent parts were generated dynamically. Drag-and-drop layout definition and resources with automatically generated default display names were effective. In some cases, it would have been useful to include questions related to several feature individuals to a single end-user interface question form. Ability to hide some alternatives dynamically instead of graying them would also have been useful.

Some useful basic functionalities were not implemented, which created extra effort and potential sources of error. For example, lack of inheritance of layouts and resources caused extra work, and editing price lists and calculations as XML was considered error-prone and not very convenient.

Empirical evaluation of performance

Performance testing of configurators is essential, as the complexity class of a configuration task is at least NP-complete in most formalisms, including the one in this work (Mackworth, 1977; Soininen, Gelle, & Niemelä, 1999; Soininen, 2000; Soininen et al., 2001; Syrjänen, 2000). A method for empirical performance testing of configurators was presented and applied to four real-world products (Tiihonen et al., 2002). The method allows performance test-

Table 5. Main characterizations of the configuration models (Adapted from II and Tiihonen, 2010).

Model	Total types	Abstract types	Concrete types	Subtypes	% as subtypes	Questions	% root questions	Constraints	Price	Effective attributes	% inherited	Effective subfeatures	% inherited subfeatures
1 C FM	9	2	7	4	44	31	58	17	adv	27	22	4	50
2 C FM sc	9	2	7	4	44	31	58	17	adv	27	22	4	50
3 C FS	3	0	3	0	0	24	88	14	adv	23	0	1	0
4 C FX	1	0	1	0	0	20	100	23	adv	20	0	0	-
5 C FL	9	2	7	4	44	28	64	13	no	24	17	4	50
6 C M	3	0	3	0	0	23	91	14	no	22	0	1	0
7 KO old	5	0	5	0	0	28	79	13	no	26	0	2	0
8 KO new	15	3	12	7	47	77	4	1	no	58	81	19	47
9 Bed	31	8	23	27	87	34	76	10	bas	31	0	3	0
10 Fireplace	7	1	6	4	57	4	75	0	no	2	0	2	0
11 Pasi	5	1	4	2	40	79	95	13	no	77	3	2	0
12 Dental	64	11	53	43	67	109	3	36	no	76	70	33	79
13 X-ray	11	2	9	4	36	37	41	3	no	32	44	5	40
14 Vehicle	28	4	24	9	32	24	75	7	bas	8	0	16	0
15 Ins 1	8	2	6	5	63	30	20	4	no	20	10	10	0
16 Ins 2	62	13	49	56	90	49	20	0	no	19	58	30	27
17 Ins 3	11	3	8	5	45	41	29	14	no	29	0	12	0
18 Ins 4	37	11	26	34	92	242	5	84	no	189	26	53	51
19 Mob 1	4	0	4	0	0	18	56	6	bas	15	0	3	0
20 Mob 2	39	9	30	38	97	65	25	28	bas	52	29	13	0
21 Mob 3	5	1	4	3	60	21	38	6	no	20	60	1	0
22 Broad	66	15	51	64	97	485	1	43	no	453	89	32	6
23 Linux	626	1	625	624	100	4369	14	2380	no	3745	67	624	0
24 Iced	8	2	6	5	63	4	75	3	bas	2	0	2	0
25 Wcar	6	1	5	2	33	10	60	3	bas	8	25	2	0
26 CarDis	10	2	8	5	50	12	58	3	bas	9	22	3	0
Total	1082	96	986	949		5985		2755		5014		881	
Total no Linux	456	95	361	325		1526		375		1269		257	
Average	18	4	14	13	48	227	50	106		193	25	34	16
Avg. no Linux	24	5	19	18	59	61	52	15		51	23	10	17
Median	9	2	7	5	46	31	58	13		25	19	4	0
Min	1	0	1	0	0	4	1	0		2	0	0	0
Max	626	15	625	624	100	4369	100	2380		3745	89	624	79

ing of configurators in terms of execution time and *using real-world configuration models with random requirements*. The test method simulates a naïve user requiring attribute values or particular subfeature realizations. Test cases with a varying number of requirements were generated following this idea. Performance tests using the method were performed with four first-modeled real-world products, a vehicle, and three compressors. For each configuration model, we generated 100 test cases with 2 requirements, 100 test cases with 4 requirements, and so on, up to the total number of questions in each configuration model.

Table 6 shows a representative example of the results: performance in the context of the largest tested compressor model. We evaluated the performance of finding both one and all configurations that satisfy the requirements. Each row lists the number of requirements (“#req”) and the number of satisfiable cases (“#sat”). Note that the sum of satisfiable and unsatisfiable cases is 100. “Find first” gives the average smodels duration of finding one configuration that satisfies the requirements, and “Unsat” gives the average smodels duration to determine unsatisfiability. “Find all” gives the average number of configurations per satisfiable case (“#cfgs/case”) and the average rate of configurations found per second (“#cfgs/s”).

Table 6. ESVS compressor results with test cases (II, Tiihonen et al., 2002).

1 Compr FM		Find all			
#req	#sat	Find first (s)	#cfgs / case	#cfgs / s	Unsat (s)
2	89	0,37	189441067	88238	0,30
4	61	0,35	18987439	76849	0,28
6	25	0,34	2234799	72687	0,29
8	9	0,33	211432	19957	0,28
10	4	0,31	1920	263	0,29
12	1	0,32	15552	526	0,29
14-28	0	-	-	-	0,30

We obtained additional performance evaluation by configuring all the characterized products using the WeCoTin user interface (Linux only partially) with a 2.4 GHz Intel Core 2 Duo laptop. All configuration models had a feeling of instant response, except the “Broadband” model’s response time was slightly more than 3 seconds before an attribute with 436 possible values was specified, after which the response time decreased to less than a second. Linux was too slow to be usable. Also, the compilation time from PCML to WCRL and then to BCRL was very satisfactory: a script that compiled all the characterized configuration models, except Linux, and a few additional test and sample models ran in 32 seconds. The results indicated good performance, and no phase transition behavior with an increasing number of requirements was found.

Utility and efficacy of WeCoTin

The PCML modeling language of WeCoTin allows efficient modeling of products for web-based sales configuration and seems suitable for engineers without background in programming or artificial intelligence. The amount of work required to create a configuration model depended to a large extent on the knowledge acquisition and validation work. The convenience of graphical constraint modeling, without the need to remember the PCML syntax and without typing in element names or values, was considered a valuable asset in modeling.

Systematic testing and ad-hoc results indicate adequate performance with the case products. There were no test cases with repeatable significantly inferior performance. In addition, there was no significant change of performance as a function of the number of requirements. The average configurations per second results show weakening with an increasing number of requirements.

This weakening seems, however, to be mostly illusory, because the number of configurations with many requirements is small—smodels duration comes mostly from reading of the BCRL program and from setting up of the computation. The Smodels inference engine appears to be efficient enough for practical use.

No critically constrained problems were found, and no phase transition behavior was apparent. As expected, the number of configurations seems to decrease exponentially as the number of requirements increases. Minor exceptions due to random requirements were encountered.

In summary, the capabilities of PCML or WeCoTin did not limit the scope of modeling, and the PCML concepts were adequate for modeling the case products. On the basis of acquired experiences, WeCoTin is suitable for e-commerce. It still lacks, however, full field testing in the form of commercial deployments.

The author of this thesis considers WeCoTin to be a major Design Science contribution. Related minor contributions are PCML, the method for systematic-configurator run-time performance testing, and the method for configuration model characterization.

7.1.3 Recommendation of configurable offerings

To demonstrate the usefulness of recommendation technologies in the context of configuration, a prototype configurator system with recommendation functionality was constructed (IV, Felfernig et al., 2010). The system `RECOMOBILE` supported personalized configuration of mobile subscriptions and the selection of a bundled mobile phone. Recommendation functionality was integrated into `RECOMOBILE` by determining dynamically default feature values.

Utility of feature value recommendation in the context of configurable offerings was shown with a user study. $N = 546$ users (mostly students, no real customers) from Austria, Finland, and Italy used `RECOMOBILE`. We compared personalized versions of the `RECOMOBILE` configurator (feature value recommendation provided) with otherwise identical versions without feature value recommendation (static default feature values). Configurations created without recommendation support were used as the basis for deriving recommendations for feature value recommendations.

Recommendations of personalized `RECOMOBILE` versions were determined with nearest-neighbor or extended naïve Bayes voter algorithms. The feature value with the highest naïve Bayes predictor or the feature value of the nearest neighbor was recommended. Only feature values consistent with the user's current feature values were recommended.

First, the user specified values for some features that characterized needs such as usage for Internet access, photography, or the form factor of the phone (see Figure 12). When the user entered a later question page, the default values of features were determined by application of the recommendation algorithms (for personalized versions), or static defaults were provided (non-personalized versions).

After a configuration session, the user answered to a number of evaluation questions related to hypotheses H1 to H8 on an 11-point Likert scale (see Table 7). When the evaluation results of users were compared with and without feature recommendation support, it was possible to show statistical significance (Student's T-test, one-tailed distribution, two-sample equal variance T-Test, see Table 8) of three hypotheses (H2, H4, and H7), marked with an asterisk in Table 7 and Table 8. The users

- (1) were significantly ($p < 0.05$) more satisfied with the overall quality of the configuration process,
- (2) perceived that the quality of the system in terms of support for finding the best options was significantly higher ($p < 0.05$), and
- (3) had their expectations regarding the solution better fulfilled with the personalized versions ($p < 0.05$).

For other hypotheses H1, H3, H5, H6, and H8 tendencies toward benefits of recommendation support were identified, but significance values stayed above 0.05 (but below 0.2). We expected that the average interaction time per page would be lower with personalized versions (H9) than with non-personalized versions. Surprisingly, the opposite was true ($p < 0.17$). A possible explanation is that users with recommendation support invested more time in evaluating the alternatives because they became more interested in the offered alternatives than users without recommendation support. This could have led to better solutions (H7).

Table 7. Overview of hypotheses H1..H9 (Felfernig et al., 2010).

Id	Hypothesis Evaluation question
H1	Personalized configurators increase a user's confidence in his or her product decision How confident are you in having selected the most suitable phone and subscription?
*H2	Users of a personalized configurator are more satisfied with the quality of the configuration process How satisfied were you with the overall quality of the selection process?
H3	Personalized configurators increase a user's trust in the presented configuration solution How high is your degree of trust in the recommendations given by the system?
*H4	Personalized configurators better support users in finding the best options How do you estimate the quality of the system in terms of supporting you in finding the best options?
H5	The probability of reusing the configurator is higher with personalized versions How high is the probability that you would use the system again?
H6	The probability of recommending the configurator to other users is higher with personalized versions How high is the probability that you would recommend the system to another user?
*H7	A user's expectations regarding the solution are better fulfilled with the personalized versions Does the combination of phone and subscription options fulfill your expectations?
H8	Personalized configurators trigger a higher purchase probability than non-personalized ones Assume that you need a new phone. How high is the probability that you would purchase the selected mobile phone?
H9	The average interaction time per page is lower with personalized versions

The image shows a screenshot of the RecoMobile customer requirements form. At the top, it says "Welcome to RecoMobile" and "RecoMobile is a Recommender System for Mobile Phones and Subscription features, which makes suggestions that are tailored to your individual needs." Below this, there is a "progress bar" with four stages: Needs, Subscription, Privacy, and Phone. The "Needs" stage is currently active. The form contains several questions with radio button options:

- Which phone style do you prefer?**
 - No preference
 - [Image of a small phone]
 - [Image of a medium phone]
 - [Image of a large phone]
 - [Image of a flip phone]
- Which phone size do you prefer?**
 - No preference
 - Tiny
 - Small
 - Compact
- Do you want to use your phone to listen to Music?**
 - No
 - Occasionally
 - Frequently
- Do you want to use your phone to take photos?**
 - No
 - Occasionally
 - Frequently
- Do you want to use your phone to access internet?**
 - No
 - Occasionally
 - Frequently

At the bottom, there is a "NEXT >>" button. Annotations in blue boxes point to various parts of the form: "entry page" points to the top section, "questions & default proposals" points to the "Which phone size do you prefer?" question, and "navigation" points to the "NEXT >>" button.

Figure 12. RECOMOBILE customer requirements, adapted from (Felfernig et al., 2010).

Table 8. Evaluation results for hypotheses $H_1..H_9$, adapted from Felfernig et al., (2010).

Id	Non-personalized	Personalized	95% significance? (p value)
H_1	5.33 (2.94)	5.73 (2.65)	no (p = 0.181)
* H_2	5.57 (2.0)	6.31 (2.19)	yes (p = 0.0184)
H_3	4.83 (2.56)	5.20 (2.45)	no (p = 0.178)
* H_4	5.05 (2.68)	5.74 (2.29)	yes (p = 0.0323)
H_5	4.43 (3.11)	5.07 (2.88)	no (p = 0.0853)
H_6	4.38 (2.99)	5.05 (2.90)	no (p = 0.0765)
* H_7	4.67 (2.33)	5.46 (2.65)	yes (p = 0.0306)
H_8	4.24 (3.44)	4.73 (3.02)	no (p = 0.162)
H_9	3.0 min. (1.05)	3.27 min. (1.67)	no (p = 0.154)

Results of this study show that configuration technologies can indeed support users in their configuration task—the recommendation experiment RECOMOBILE indicated the usefulness of recommendation technologies.

The author of this thesis considers the proposed extended recommendation algorithms for feature value recommendation to be Design Science contributions. They can potentially provide more relevant recommendations than the original algorithms. Confirming the utility of the recommendation-supported configuration approach is an additional contribution.

7.1.4 Relation between services and the configuration approach

The services branch of research provided insights that the author of this work is not aware of in previous work, which offers minor contributions:

- Analysis of offered variation of case service offerings and identification of specific sources of offered variation in services based on a formal relationship.
- Recognition that the required scope of service configuration modeling is broader than in modeling of the offered variation of physical products (process, stakeholders and their properties).
- Recognition that reconfiguration seems to be more important in the configuring of contract-based services than in that of most physical products.
- Verification that basic service contract sales configuration with configurators designed for physical products is possible, but a conceptual mismatch in modeling is possible too.
- Recognition that reconfiguration, process modeling, and management of several non-commensurate prices may be required to fully support service contract configuration.

7.2 Sales Configurator Information Systems Design Theory

According to Gregor (2006), a recipe-like *prescription* exists when theory enables an artifact to be constructed by describing a method or structure for its construction. Gregor and Jones (2007) further refine the idea into *elements of information system theory*. They have identified 8 components; see Table 9.

Table 9. Components of Information Systems Design Theory (Gregor & Jones, 2007, p. 322).

Component	Description
Core components	
1) Purpose and scope	"What the system is for," the set of meta-requirements or goals that specifies the type of artifact to which the theory applies and in conjunction also defines the scope, or boundaries, of the theory.
2) Constructs	Representations of the entities of interest in the theory.
3) Principle of form and function	The abstract "blueprint" or architecture that describes an IS artifact, either product or method / intervention.
4) Artifact mutability	The changes in state of the artifact anticipated in the theory, that is, what degree of artifact change is encompassed by the theory.
5) Testable propositions	Truth statements about the design theory.
6) Justificatory knowledge	The underlying knowledge or theory from the natural or social or design sciences that gives a basis and explanation for the design (kernel theories).
Additional components	
7) Principles of implementation	A description of processes for implementing the theory (either product or method) in specific contexts.
8) Expository instantiation	A physical implementation of the artifact that can assist in representing the theory both as an expository device and for purposes of testing.

In this work, a partial prescription for a sales configurator was proposed; we call it *sales configurator information systems design theory* (SCISDT). Table 10 summarizes WeCoTin in this framework, and the following sub-subsections discuss each component. SCISDT is partial, because it does not address all aspects of a sales configurator. SCISDT also applies other contributions as subsystems, such as Smodels.

7.2.1 Purpose and scope

Companies with a mass-customization strategy need to provide choice navigation capability (Salvador et al., 2009). Configurators are the primary means to this end. In the scope of this work, generic configurators, aka *configuration toolkits*, enable the creation of configurator instantiations for individual companies or product lines. Configurators can provide numerous other benefits. On the other hand, taking a configurator into use, and operating and keeping it up to date, also incurs significant costs; the total cost of configurator ownership should be justifiable.

Although there are numerous individual configurator instantiations and generic-purpose configurators that enable such instantiations to be created, it was deemed that none met all the desirable properties that we considered important: A (sales) configurator should

- be easy to set up without programming,
- enable fluent product modeling of products based on a well-founded high-level modeling conceptualization, and
- be easy to maintain.

In addition, we wanted to experiment with applying a novel logic-based method for problem solving that would enable high-level configuration modeling and consistent and complete inference.

Table 10. Components of Sales Configurator Information Systems Design Theory (SCISDT).

Core component	Description (as explicated by WeCoTin)
1) Purpose and scope	A web-based sales configurator that fulfills a set of major requirements identified in Section 4.1.
2) Constructs	Concepts of configuration knowledge, product configuration modeling language PCML, weight constraint rule language.
3) Principle of form and function	A high-level architecture and main functions of components was presented along with main working principles (Il, Anderson & Pasanen, 2003; Tiihonen et al., 2003; Tiihonen & Anderson, 2005)
4) Artifact mutability	WeCoTin has several internal interfaces that enable replacement of major components. It has also been designed to be flexible in numerous aspects, such as different ways to determine prices, and support for several languages.
5) Testable propositions	The main propositions were capability to model and configure real products. Another proposition is adequate performance. These aspects were tested with highly satisfactory results.
6) Justificatory knowledge	The modeling constructs of PCML were given clear formal semantics by mapping them to the weight constraint rule language. This mapping also enables sound and complete inference by the Smodels system. Additional practical justification is provided by the expository instantiation (element 8 below.)
7) Principles of implementation	Steps to apply WeCoTin have been provided (Tiihonen & Anderson, 2005). Successful implementation requires also the other core capabilities of mass customization: solution space development and robust process (Salvador et al., 2009).
8) Expository instantiation	WeCoTin sales configurator and augmenting 26 configuration models.

7.2.2 Constructs

ISDT *constructs* represent the entities that are of interest in the theory, and corresponding terms should be defined as clearly as possible (Gregor & Jones, 2007).

In the context of this work, it is somewhat challenging to draw the line between the constructs and principles of form and function. Relevant ISDT constructs include at least the conceptualization of configuration knowledge, and

object-oriented product configuration modeling language (PCML). A sales configurator (WeCoTin) as a whole and its major parts (Modeling Tool, Configuration Tool) also belong to the relevant ISDT constructs.²¹ Underlying these as subsystems are Smodels, the weight constraint rule language (WCRL), and the method of translating configuration knowledge to WCRL. These underlying subsystems were developed outside the scope of this work.

It is noteworthy that the conceptualization was constructed in such a way that it retains the natural thinking patterns used in companies to describe the variation of product families. Compositional structure of products and configurable attributes are the main mechanisms for capturing offered variation. Taxonomy with inheritance generalizes the approach. The full conceptualization also supports connection-oriented concepts and resources that have proven to be useful in earlier work. All these can be given formal semantics by mapping them to formal constructs (see Section 7.2.3).

7.2.3 Principle of form and function

Principles of form and function “define the structure, organization, and functioning of the design product or design method. The shape of a design product is seen in the properties, functions, features, or attributes that the product possesses when constructed” (Gregor & Jones, 2007, p. 325).

A configurator should have separate environments for the modelers and end users—the concerns are separate. Nevertheless, WeCoTin offers the modeler the capability to rapidly test the created or edited configuration model.

WeCoTin was built on a layered architecture. We propose this as a significant principle of configurator construction. This provided a clear separation of

- formal inference, which in this case is logic-based;
- high-level modeling concepts, which can be provided with formal semantics and automatically mapped to a form suitable for inference; and
- the end-user interface, which does not require programming—in this case generation of the user interface is facilitated by the high-level modeling language.

The main functions of a configurator include checking for the consistency and completeness of a configuration, with the capability to prevent from ordering a product based on a configuration that does not meet these criteria. Price is an integral element that must be managed within the scope of a configuration task.

A hierarchy of modeling languages is also present: the high-level configuration modeling language (PCML) is aimed to be adequate for modelers. This is compiled into a formal weight constraint rule language with variables. Finally, WCRL is compiled into a simple basic constraint rule language without variables.

²¹ We consider these instantiations as ISDT constructs because they are concepts of interest in SCISDT. They are not constructs in the terminology of (Hevner et al., 2004) applied elsewhere in this thesis.

The view of the author of this thesis is that future configurators should support recommendation functionality to support users with choice navigation. Case-based recommendation approaches investigated in this work seem to be potentially viable, but further research is required. In addition, future configurators should provide recently identified user support capabilities to avoid the product variety paradox where increased offered variety may decrease sales volume (Trentin, Perin, & Forza, 2013). These capabilities are focused navigation, flexible navigation, easy comparison, benefit-cost communication, and user-friendly product-space description capabilities (Trentin et al., 2013)(Trentin et al., 2013).

7.2.4 Artifact mutability

WeCoTin has several internal interfaces that enable replacement of major components. For example, it is expected that Smodels could be relatively easily replaced with another inference engine based on weight constraint rules. There are interfaces for configuration model manipulation and manipulation of configurations. These make it easier to create different modeling environments and user interfaces for end users.

WeCoTin has also been designed to be flexible in numerous respects, such as different ways to determine prices, and built-in support for several end-user languages and tax models. Product changes do not require programming changes in the user interface for end users: a template gives the general visual appearance, and WeCoTin generates the product-specific part (the modeler can change the input control types and determine their sequencing).

But architectural mutability and suitability for generic tasks including dimensioning and connections could potentially be higher. Generic dimensioning tasks would require integrating additional inference or calculation mechanisms; user-specified connections would require appropriate user interface support. In some configuration tasks, a dynamically determined flow of the configuration process based on previous answers would be necessary. There are no specific provisions for these needs.

7.2.5 Testable propositions

The main propositions were capability to model and configure real products and adequate performance in this context. These aspects were tested with highly satisfactory results.

Using WCRL and Smodels to provide inference seems to be a feasible proposition for building a sales configurator. The typical approach in previous work has been based on constraint satisfaction.

7.2.6 Justificatory knowledge

The configuration knowledge conceptualization is based on a synthesis of previous work and additional experiences from interviews in ten companies and

two case studies (Soininen & Tiihonen, 1995; Tiihonen, 1994; Tiihonen, 1999; Tiihonen & Soininen, 1997a); see also research unit 1 in O, p. 19.

PCML allows the offered variation of products to be expressed on a high level that product experts can understand. Furthermore, the modeling concepts of PCML were given clear formal semantics by being mapped to a weight constraint rule language. This mapping enables sound and complete inference by the Smodels system, giving a foundation to the claim that, if a sales configurator is built on such well-founded principles, a working sales configurator can be implemented.

7.2.7 Principles of implementation

We omit to address this optional component of an ISDT with respect to the construction of generic configurators. Providing sound principles might require several applications of the SCISDT.

To produce individual configurator instantiations, one should bear in mind that implementing a configurator is not just an IT project. Additional existing capabilities are required. These include other core capabilities of mass customization, namely solution space development and robust process as identified by Salvador et al. (2009). Concrete steps to apply WeCoTin have also been provided (Tiihonen & Anderson, 2005).

7.2.8 Expository instantiation

The WeCoTin sales configurator and the numerous configuration models work as an expository instantiation of the SCISDT. WeCoTin exhibited good performance and was capable of modeling and configuring all the case products.

As side results, new methods of characterizing configuration models and measuring configurator performance were developed.

Numerous configuration models based on the offered variation of real offerings were developed. These show how WeCoTin could be applied in respective companies to provide choice navigation support.

7.2.9 Identification of contributions

The author considers SCISDT (and expository instantiation WeCoTin) to be the most significant contributions of this thesis. As a side result, it was shown that configuration knowledge translated into a form of logic programs (Soininen, 2000; Soininen et al., 2001) can act as a basis for a configurator and that inference based on the stable-model semantics of logic programs can indeed provide a basis for constructing a practically applicable sales configurator.

7.3 This work and the guidelines of Design Science

Full discussion about the relation of this work and the guidelines of Design Science research (Hevner et al., 2004) is omitted, because most relevant aspects have already been addressed by the evaluation via the ISDT approach. A

summary of this work with respect to “Design-Science Research Guidelines” (Hevner et al., 2004) is presented in Table 11.

Table 11. This work versus “Design-Science Research Guidelines” of Hevner et al. (2004).

Guideline	Description
#1: Design as an Artifact	Viable artifacts (WeCoTin configurator, configuration conceptualization, recommendation algorithms) have been produced.
#2: Problem Relevance	The business relevance of sales configuration was identified in the Introduction and Previous Work sections of this work.
#3: Design Evaluation	The utility and efficacy of the artifacts were discussed above. In addition, a partial information systems design theory (SCISDT) was proposed.
#4: Research Contributions	The author of this work claims that clear contributions exist. An outline of the contributions of this work was identified in Section 1.3.2 (Figure 4, p. 19). Sections 7.1 and 7.2 discuss the contributions with more details.
#5: Research Rigor	Relevant applicable knowledge from the knowledge base has been sought and selectively taken into account in the production of the artifacts. Performing systematic literature reviews could have revealed some information in the knowledge base that was omitted. Adequate Design Science methods have been applied.
#6: Design as a Search Process	The development process was stepwise: often a limited baseline of functionality was developed first, and functional extensions were developed, prioritized by modeling experiments and heuristic “touch” on the importance of requirements. In some cases, design-test iterations were performed to produce satisfactory functionality. For example, significant aspects of the user interface for end users were re-designed and re-implemented after usability tests (Talja, 2006), and the “advanced calculation mechanism” (Nurmilaakso, 2004) was added.
#7: Communication of Research	Publications annexed to this work exhibit scientific, mainly technical communication with some managerial insights. But managerial and professional communication could have been more extensive. In addition, more ambitious selection of publication forums might have been appropriate.

8. Discussion

This section begins with a comparison with related work (Section 8.1), followed by discussion on the threats of validity (Section 8.2). Next, answers to the research questions are explicated (Section 8.3). Finally, avenues for future research are identified (Section 8.4).

8.1 Related work

8.1.1 Conceptualizations for Configuration knowledge

A similar synthesis as publication I, based on a representation that employs Unified Modeling Language (UML) (Rumbaugh et al., 1999) with specific stereotypes and Object Constraint Language (OCL) (Warmer & Kleppe, 2003), was proposed for modeling configuration knowledge (Felfernig, Friedrich, & Jannach, 2000a; Felfernig et al., 2000b; Felfernig et al., 2002; Felfernig, Friedrich, Jannach, Stumptner, & Zanker, 2003; Felfernig, 2007). The stereotypes include the connection-oriented and resource-oriented concepts along with a taxonomical hierarchy of component types (Felfernig et al., 2000a; Felfernig et al., 2000b; Felfernig et al., 2002; Felfernig, 2007). The benefits of UML include its being widely known in the software industry. Publication I did not specify a syntax for constraints. On the other hand, OCL is potentially difficult for modelers. Many details of the concepts synthesized in publication I have been omitted from the UML-based approach.

8.1.2 Logic-based configurators

The author is not aware of previous explicitly presented information system design theories in the domain of configurators. Previous work includes configurators applying numerous problem-solving methods (see Section 2.4.2). Here we concentrate on logic-based systems, because they are a major ingredient of SCISDT. Previously weight constraint rules were applied directly to model individual configuration problems (Syrjänen, 2000). WeCoTin differs by the general purpose, multi-domain approach, and high-level configuration of specific modeling concepts with corresponding tools for modeling.

Numerous other logic-based methods have been applied in configurators. Description logic has been applied (McGuinness & Wright, 1998b; Wright et al., 1993; Wright et al., 1995). It is not entirely clear, however, what role de-

scription logics play as an inference engine for supporting the actual configuration task. According to McGuinness & Wright (1998b), the system was used to deduce logical implications (deductive closure) of the basis user inputs. According to Wright et al. (1995), this system has been augmented in the case of larger products with external search algorithms and special-purpose algorithms—description logic was used to check the integrity of the results.

Constraint logic programming (CLP) has been applied for configuration problems (Sharma & Colomb, 1998). The implementation was based on the ECLiPSe constraint logic programming environment using the Finite Domains (FD) library. In this implementation, a high-level language was developed to model configuration knowledge, and it was based on concepts similar to a subset of publication I. The language included taxonomical hierarchy of component types and their attributes, ports, and part relationships. Configuration models expressed in this language were then translated into CLP programs, which enabled configuration based on expressed requirements. Results indicative of adequate performance in automatic configuration were achieved. Sharma and Colomb’s work bears many similarities to the approach of this work. Detailed comparison, however, is difficult for lack of details. We go further with published empirical evaluation of the system and efficacy of modeling. Furthermore, we have provided a partial ISDT (SCISDT) for sales configurators.

As in WeCoTin, weight constraint rules are applied in the OOASP framework (Schenner et al., 2013) in order to combine high-level object-oriented configuration modeling with an underlying inference system that follows the answer set programming (ASP) paradigm. OOASP is an experimental framework that consists of a set of logic program files (*.lp). It can provide inference for a number of use cases: checking a configuration, completing a configuration, reconfiguration “reconciliation,” and choosing the best knowledge base for reconciliation. The latter two use cases are not supported by WeCoTin. Furthermore, OOASP supports arbitrary associations that make it possible, for example, to define product topology or compositional structure. In contrast, the only association type of PCML/WeCoTin defines the compositional structure. On the other hand, OOASP lacks user interfaces for modelers and end users, and the modeler should be proficient in weight constraint rule language. Furthermore, OOASP has been evaluated only as a demonstrator of the ASP approach.

Many (if not most) configurators infer the consequences of configuration decisions and check consistency during the configuration process. WeCoTin checks “behind the scenes” during the configuration process that there exists a way to complete the configuration—that is, it detects dead ends. We are not aware of other configurators that provide this functionality.

8.1.3 Configuration modeling and performance testing

When configuration models are characterized in previous work, usually the number of component types and/or connections is specified (Fleischanderl et

al., 1998; Mailharro, 1998; Sharma & Colomb, 1998; Syrjänen, 2000). We are not aware of previous work with a deeper characterization of configuration models, such as application of inheritance, characterization of compositional structure, and other modeling mechanisms.

We are only aware of limited configurator performance testing in previous work; Syrjänen (2000) configured the main distribution of Debian GNU/Linux using configuration models expressed by an extension of normal logic programs. Syrjänen's approach seems to perform better than ours, as the Debian configuration models are substantially larger. But modeling in this work was performed on a higher-level conceptualization that did not offer opportunities for the manual tweaking of performance. Other previous performance and modeling experiments have not provided enough details to allow detailed performance comparison. These include Sharma and Colomb (1998) and Mailharro (1998). Sharma and Colomb (1998) configured thin Ethernet cabling with developed constraint-logic-based language. Mailharro (1998) used the Ilog system to configure the instrumentation and control hardware and software of nuclear power plants. Syrjänen and Mailharro applied larger products for testing than the systematically tested products of this work. On the other hand, performance testing in this work differs from previous work in that it applied configuration models of several products and varying numbers of random requirements that could reveal phase transition behavior.

The CLib configuration benchmarks library contains 21 configuration problems (Subbarayan, 2005). These benchmarks include the systematically tested cases of this work. We selected to test performance only on our "own" models whose semantics were known and that were expressed in a form that facilitated PCML modeling with the original problem structure. This is not the case for problems expressed in conjunctive normal form, such as those integrated into CLib based on the idea of Sinz, Kaiser, & Kuchlin, (2003).

8.1.4 Recommendation of configurable offerings

Falkner et al. (2011) provided an overview of recommendation technologies for configurable offerings with broader scope than this work. They provided an overview of recommendation (1) of features (determine which features to exclude from being asked from the user, or rank features to determine which features to configure next), (2) of explanations (in case of conflicting requirements, recommend (minimal) sets of requirements that must be changed or deleted for a solution to be found), and (3) of feature values. Of these, the author of this work sees (1) and (2) as important topics, but outside the scope of this work. With respect to (3), the overview of publication III is more detailed than the work of Falkner et al. (2011). Publication IV provides some empirical evaluation, which is absent from Falkner et al. (2011).

The CAWICOMS Workbench (Ardissono et al., 2003) had utility-based recommendation support. The interests and skills of users were identified on several dimensions and represented in a user model. For example, in the context of telecommunications equipment, interest in product reliability could be a dimension. Persons interested in high product reliability would get recom-

mentations on options that enhanced reliability. For less-skilled users, questions related to some features would not be shown, and, e.g., dynamic defaults would be applied. In comparison, the CAWICOMS approach is more dynamic than the approach of publication IV because of its totally dynamic user interface, and its recommendation support extends to the recommendation of features (questions). On the other hand, the case-based approach of III and IV can potentially work with less knowledge engineering effort. Furthermore, this work provided evaluation with users (IV).

VITA financial-service recommenders identify combinations of financial products to create a portfolio that matches the needs of the individual customer or the family (Felfernig et al., 2007). Here, the configuration task is usually simple in terms of compatibility, but finding a combination of suitable financial products may be challenging. The VITA financial-service recommenders (Felfernig et al., 2007) are based on the MAUT approach, and they have been applied in production use. Stolze & Field (2000) help users select suitable insurance products from a catalog of fixed products or products with limited configurability, such as the deductible in case of insurance products. They use a scoring system wherein the user selects affecting factors and explicitly gives importance values. This calculation seems to be a variation of the MAUT approach. The idea is to find a suitable solution from the offerings of different providers.

The proposed algorithms build on the ideas of Cöster et al. (2002). This work goes further because it provides empirical validation. Missing validation is common in previous work—we were unable to find evaluations of performance of recommendation systems in the context of configurable offerings.

Cunningham et al. (2001) described a system that supports selecting a base product to configure with recommendation technologies. Actual configuration is supported with “ordinary” configuration techniques. No evaluation with users was provided. A similar WeCoTin extension, “CCCP—a Tool for Comparing Configurable Products” was developed (Heiskala, Anderson, Huhtinen, Tiihonen, & Martio, 2003).

Zanker, Aschinger, and Jessenitschnig applied direct constraint-based modeling for planning personalized tourist agendas with flights, accommodation, and activities (2010). Again, no evaluation was provided. The aim was also different: to calculate a small number of substantially differing bundles for the user to select from.

8.1.5 Service configuration

The idea of service configuration can be considered relatively well known in the literature, as discussed in Section 2.6.2. Explicit discussion on service configuration, however, is relatively scant. Heiskala et al. (2005) presented a conceptual analysis of whether the benefits and challenges of mass customization, configuration, and configurators are relevant in service settings, taking the IHIP characteristics as the basis of analysis.

This work indicated that is often necessary to model the customer, related equipment, other stakeholders, and properties of all of these. A similar view has been expressed by several authors (Dausch & Hsu, 2006; Ma, Tseng, & Yen, 2002; Stolze & Field, 2000; Wimmer, Mehlaui, & Klein, 2003; Winter, 2001).

The view of the author of this work differs from that of Winter (2001), who said, “While maximum flexibility has been the foundation of product models in mechanical engineering, service product variants primarily reflect regulations, pricing rules, customer properties, or risk properties. Therefore, more constraints have to be represented in general” (p. 205). The cases of this work required a relatively small number of constraints.

The pricing models encountered in this work were compliant with those identified in previous work by Lovelock (1983) and de Miranda et al. (2006).

8.2 Threats of validity

According to (Bryman & Bell, 2007, p.42), *validity* of research is concerned with the integrity of conclusions. Shadish, Cook, and Campbell (2002) are more specific: they emphasize the role of inference as the subject of validity and define validity as “the truth of, correctness of, or degree of support for an inference” (p. 513). They distinguish four main types of validity. Next to be addressed here are plausible threats to the main types of validity.

Statistical conclusion validity

Statistical conclusion validity concerns “the validity of inferences about the correlation (covariation) between treatment and outcome” (Shadish et al., 2002, p. 38). In this thesis, it pertains only to the evaluation of the utility of the recommendation approach, specifically to the performed tests of hypotheses H1 .. H9. The first author of publication IV verified that assumptions of the applied test (Student’s T-test, one-tailed distribution, two-sample equal variance T-Test) were not violated. The participants of the on-line user study were mostly recruited from the students and faculty of four universities via e-mail and bulletin board advertisements. This is likely to introduce sampling bias with respect to the age and educational background; possibly also the distributions of profession, gender, and other background factors are biased. The statistical significance of three hypotheses was confirmed at 95% level of confidence, and other hypotheses H1..H8 could have been confirmed only at 80% level of confidence ($0.0765 < p < 0.178$), which is not sufficient. However, all these hypotheses exhibit a tendency to the direction of benefit provided by recommendation technology. This makes it less likely that the hypotheses would have been confirmed as a result of random chance caused by the testing of multiple hypotheses, known as the *multiple comparisons problem*. Configuration sessions with personalized configurators took longer ($p = 0.154$, H9), which was contrary to the original assumption. As a whole, it seems safe to claim that recommendation technology can provide benefits to users.

Construct validity

Construct²² validity concerns “the validity of inferences about the higher order constructs that represent sampling particulars” (Shadish et al., 2002, p. 38). A plausible threat to construct validity comes from conceptualization of compositional structure. Compositional structure was conceptualized through *PART DEFINITIONS* (aka subfeature definitions) with semantics that a valid individual of the whole type has the number of part individuals specified by the *cardinality* as parts with the specified *part name*. Each individual as a part must be of one of the *possible part types*. These concepts may match neither those of practitioners nor those of other researchers. Numerous products were modeled, however, and the concepts accurately model the compositional structure of configurable products with a good match to the modeling needs. It is plausible that the concepts would be simple to explain to practitioners.

Internal validity

Internal validity concerns “the validity of inferences about whether observed covariation between A (the presumed treatment) and B (the presumed outcome) reflects a causal relationship from A to B as those variables were manipulated or measured” (Shadish et al., 2002, p. 38). Internal validity is subject to author bias. Configuration modeling efficacy results are particularly problematic, because all modeling and evaluation of modeling were performed by researchers who were involved in WeCoTin development and who were used to modeling. Furthermore, the author of this work defined the configuration model used in *RECOMOBILE* recommendation experiment. It might be that some aspects that are relevant to users were omitted, which might affect the satisfaction of corresponding users.

External validity

External validity concerns “the validity of inferences about whether the cause-effect relationship holds over variation in persons, settings, treatment variables, and measurement variables” (Shadish et al., 2002, p. 38). The main artifact WeCoTin configurator has been evaluated with a subset of relevant views (performance, the characterization of configuration models, and applied modeling concepts). A number of aspects, however, can threaten the external validity of WeCoTin’s claimed practicality and utility. In other words, it may be the case that WeCoTin does not provide claimed utility or is not practically applicable in generalized settings. Plausible threats to the external validity are discussed next.

Case selection by convenience. The cases, ten companies with 22 products that were modeled and configured, were selected by convenience from either existing or potential research partners. The case companies are real and relevant, and represent industries that are typical in Finland: manufacturers of investment goods, especially the machine-building sector, augmented with healthcare equipment manufacturers. Each case product can be configured

²²‘Construct’ in ‘construct validity’ is should not be confused with term ‘construct’ in the sense of (Hevner et al., 2004) applied in the rest of this thesis.

with direct selections; little search is required. The modeled cases do not cover the most challenging configuration tasks, such as telecommunications networks. Generalizing the claims of applicability of WeCoTin to other types of companies or products requires careful judgment.

Configuration modeling may require modeling expertise. All modeling was performed by researchers who were involved in configuration topics and WeCoTin development. Modeling expertise could be needed to model products with WeCoTin—at least, the efficacy of modeling is subject to modeler background.

Incomplete validation of the configuration models. Five configuration models from three companies were test-used by company representatives. In addition, configuration demonstrations and immediately following focus groups were used to validate seven additional configuration models. Nevertheless, most configuration models were applied only as demonstrations, and modeling errors could have been left unnoticed. Many details were changed in iteration toward complete models, but such changes were minor with respect to selection and application of modeling concepts. Thus, it is reasonable to assume that a more thorough validation would not have revealed requirements that cannot be met with WeCoTin.

No real customers. No operational use of WeCoTin has taken place. Validation in a full business context with real end users is thus missing. The RECO-MOBILE experiment was not performed with real customers, which might affect the results.

Summary

Despite the identified threats to validity, the rigor of the evaluation of WeCoTin can be considered very high among evaluations of configurators. On the other hand, some commercial systems are widely applied in industrial settings, whereas WeCoTin lacks this form of validation. Applicability of the artifact to real-world problems was demonstrated, and ability to generalize exists in the sense that the configuration tasks from different industries can be supported. Nevertheless, it is easy to identify configuration tasks that cannot be fully supported.

8.3 Answers to the research questions and contributions

8.3.1 RQ1: What are the concepts central to configuration knowledge?

We consider the main concepts of publication I to be central. Publication I covers the *connection-based* (Mittal & Frayman, 1989), *resource-based* (Heinrich & Jüngst, 1991), *structure-based* (e.g., Cunis et al., 1989), and *function-based* (Najmann & Stein, 1992) approaches. It also extends the previous conceptualizations in several ways. The main concepts are *component*, *function*, *port*, and *resource*. These are treated uniformly with respect to several criteria: defined both as types and as individuals, organized in classification taxonomies, and having attribute definitions. Furthermore, the types can be specified as abstract or concrete. Component and function types can specify config-

urable compositional structure with *part definitions*, where allowed types that can realize a part, cardinality, and part name are specified.

For sales configuration of some products, a subset of the concepts is enough. We conclude, based on modeling (and configuring) without difficulties and with an acceptable conceptual match, the sales view of real-world products (14 fully, 8 partially) that the function-oriented subset of the configuration conceptualization is useful for modeling configuration knowledge. In other words, by modeling the sales configuration view of several products with WeCoTin, it is possible to narrow down the set of central concepts. The most central concepts include typed objects, their attributes, subfeature (part-) definitions, and constraints. Typed objects can be called “components,” “functions,” “features,” or “objects.” In the context of sales configuration, the author of this work prefers “feature.” It is important to have available compositional structure that distinguishes the part name (“role”) and specifies allowed types and cardinality. Inheritance and abstract types must be supported. Topological concepts and resource-based concepts, while important, are not as essential. But it is difficult to imagine effective configuration modeling of complex telecommunications equipment or computer systems without support for connection-based and resource-based concepts.

The set of concepts of publication I is not minimal in the formal sense. In the view of the author of this thesis, this is not a problem: the clarity of configuration models should not be compromised by strictly minimizing the number of concepts; the conceptualization has a good balance between minimizing the number of concepts and making configuration models understandable.

8.3.2 RQ2: How to construct a practical and computationally well-founded sales configurator?

A partial information systems design theory (ISDT) for sales configurators (SCISDT) fulfilling a set of major requirements was presented. SCISDT aims to provide a recipe-like prescription for the construction of similar artifacts. The major ingredients include a high-level object-oriented configuration modeling language that is based on a well-founded conceptualization, a method of translating configuration models into weight constraint rules, and an inference engine based on weight constraint rules. A high-level architecture and the main functions of major components were presented along with the main working principles. SCISDT is based on the design of WeCoTin, a sales configurator that supports mass customization of complex products. WeCoTin is computationally well founded because it was constructed based on the idea of translation of configuration knowledge into weight constraint rules (Soininen, 2000; Soininen et al., 2001). This principle provides theoretical grounding and allows for sound and complete inference. In addition, WeCoTin incorporates tools that allow graphical configuration modeling, semi-automatic generation of user interfaces, and several other aspects that ease long-term management. WeCoTin works as an expository instantiation of SCISDT.

The utility and efficacy of WeCoTin were evaluated. Applicability of WeCoTin and its modeling capabilities to industrial problems were evaluated; the creat-

ed configuration models were characterized in terms of size and modeling concepts that were applied. The created and evaluated sales configuration models were small, but representative of the Finnish industry. The modeling language of WeCoTin (PCML) was adequate for modeling the products. WeCoTin had demonstrably adequate performance with the four models that were systematically tested and ad-hoc manual configuration with other configuration models. An exception to this sufficient performance is the large Linux model, in which achieving sufficient performance would require at least the capability to control when full inference was performed, and possibly other optimizations.

We claim that the main artifact of this thesis, the WeCoTin configurator, is a practical sales configurator in terms of a number of key aspects. Its generalization as the SCISDT provides the “recipe” that answers RQ2.

8.3.3 RQ3: Can users be effectively supported in finding suitable products and services with personalized recommendations?

We provided an overview of the approaches to integrate recommendation of configuration settings (feature values) with configuration technologies. This integration shows potential for reducing the so-called mass confusion phenomenon (Huffman & Kahn, 1998) that prevents users from identifying products and services fitting their wishes and needs.

We proposed extensions to already existing case-based recommendation algorithms to integrate importance weights and similarity metrics with a classification approach that has not, to the knowledge of the author of this thesis, been applied in the recommendation of configurable offerings.

A basic evaluation of the case-based collaborative recommendation approach was provided by a user study in which test subjects ($N = 546$) configured mobile subscriptions and selected a phone, both with and without recommendation support. Recommendation support was provided by application of the nearest-neighbor and extended naïve Bayes voter algorithms to calculate feature value recommendations. All eight hypotheses based on user evaluation exhibited a tendency to the direction of benefit provided by recommendation technology. It was also possible to show in a statistically significant way ($p < 0.05$) that users with recommendation support (1) were more satisfied with the overall quality of the configuration process, (2) perceived that the quality of the system in terms of support for finding the best options was significantly higher, and (3) had their expectations regarding the solution better fulfilled. But the improvement was relatively small—for example, satisfaction with the configuration process increased from 5.57 to 6.31 on an 11-point Likert scale.

We believe that equipping configurators with personalized recommendation of feature values will be a major source of additional support for users performing choice navigation in the context of a mass customization strategy. Recommendation support can potentially extend the range of configurable offerings that can be configured as self-service in such scenarios as e-commerce.

For the reasons mentioned above, we give a positive answer to the research question RQ3. Future work is needed, however, to bring these functionalities to commercial systems and to generalize the approach.

8.3.4 RQ4: How does service configuration differ from the configuration of physical products?

Configuration of services was addressed in the context of service contract configuration of telecommunications, maintenance service, and insurance offerings. The offered variation of the case services contained *what*-variation defining the desired service outcome, some temporal aspects of *when*-variation, and limited *with-what*-, *how*-, and *where*-variation. Surprisingly, no *who*-variation was identified in the sense that service personnel or their skills would have been selected. Through *what*-variation, it was sometimes possible to configure the extent of customer participation. A formal relationship brought about certain sources of variation: pricing options, paying and billing, information and reporting, service quality attributes (promised performance and response time, the availability of maintained equipment), and even loyal-customer benefits. Ownership and intellectual-property rights were not configured.

The properties of a customer, equipment within the scope of service, other stakeholders (*to whom*), and the environment influenced service availability, feasible parameter values, and pricing. This extends the scope of modeling beyond the service product itself to ensure the correctness of the configuration and pricing. Furthermore, explicitly modeling service delivery processes and resources may be required to clarify the customer's role and help manage expectations. Complex customership and service provider's partner networks potentially complicate configuration modeling. If subcontractors are used, for example, the customer may be interested in who they are or what their qualifications are.

Service contract configuration sometimes required the management of several non-commensurate prices, such as initiation, periodic, and pay-per-use prices. In such services as telecommunications or insurance, the installed base of service contracts needs to be managed so that reconfiguration is possible to match a customer's changed needs. Furthermore, sometimes service processes need to be configured to deliver the configured service.

The extended scope of modeling may lead to a conceptual mismatch with terminology such as *component or feature types*. In addition, modeling configurable processes are not convenient. Despite these challenges, we conclude that at least some, and probably most, configurable service offerings can be modeled and configured with traditional configurators. Reconfiguration and management of several prices may require additional support.

8.4 Future research

Applying SCISDT independently would verify it and provide further evidence on applying weight-constraint-rule-based inference with a high-level modeling conceptualization and language.

WeCoTin could be extended. Potential extensions include repair functionality of inconsistent configurations, personalized recommendations, optimization support, enhanced ways to express requirements, visualization, improved end-user interface (with enhanced capabilities for supporting the end user to prevent the product variety paradox (Trentin et al., 2013) and dynamic aspects such as the sequence of questions depending on previous answers), and support for reconfiguration. “Syntactic sugar” could be offered: a minor inconvenience was apparent in the case of an optional subfeature (cardinality 0 to 1), and exactly one allowed type: it was difficult to invent a name for the feature type and the subfeature. For example, an optional radio would be described with a subfeature definition named `radio` and a feature type `radio`.

The proposed configuration knowledge conceptualization could be extended on many fronts. *Geometric knowledge* could be included to facilitate visualization, spatial layout design, and so on. In the context of service configuration, there is a need to model *processes* (activities and their resources). *Optimality knowledge* could guide the configuration process. Further extensions could cover *pricing, costs, and other sacrifices*. *Mixin types* (Hedin et al., 1998) could potentially offer a simpler alternative to multiple inheritance by providing the reuse of attribute specifications and constraints in main types.

Recommendation of configurable offerings provides numerous research opportunities. Nearest to this work, recommendation algorithms should be extended to cope with structurally varying configurations that make the set of features dynamic. There remains to evaluate how well recommendation algorithms perform. Furthermore, alternative ways to determine similarity of symbolic-feature values should be studied.

One opportunity is constructing a generic service configurator that has a service-specific modeling conceptualization such as the four-worlds model (Heiskala, 2005; Heiskala et al., 2005; Heiskala et al., 2006). Such a configurator would provide a good conceptual match for service configuration modeling. One interesting approach in this direction would be to apply metamodeling languages such as NIVEL (Asikainen & Männistö, 2010)—it could become straightforward to define configuration languages resembling PCML with a good conceptual match in such domains as software configuration or services.

9. Conclusions

Configurators have been a success story for artificial-intelligence techniques because of their fundamentally well-defined basic problem. This work continues on the same track, but it has taken a somewhat higher-level point of view: the primary interest has been to provide useful information systems, not actually to investigate AI technologies themselves.

By the Design Science research approach, a number of artifacts supporting sales configuration of physical products and services were developed, aiming to advance practically applicable configurators. The main artifact in the center of research was the WeCoTin sales configurator. Other artifacts contribute to the foundations of WeCoTin, to its evaluation, or to potential extensions of it.

We presented a configuration conceptualization that combined the ideas of previous approaches and extends them. Its main concepts are *component*, *function*, *port*, and *resource*. These main types are treated uniformly with respect to several criteria: defined both as types and individuals, organized in classification taxonomies, and equipped with attributes. Components and functions can have configurable compositional structure where a part definition specifies allowed types, cardinality, and a part name. We believe that we have struck a good balance between minimality and expressiveness.

Enabled by the novel principles of WeCoTin artifact, we proposed an information systems design theory (SCISDT) for sales configurators. WeCoTin consists of a graphical modeling environment Modeling Tool and a Web-based Configuration Tool that supports the configuration task. The modeling language has clear formal semantics, provided by mapping it to a form of logic programs. WeCoTin and SCISDT are based on a well-founded modeling conceptualization and a corresponding high-level object- and feature-oriented modeling language with clear formal semantics, which are provided by the modeling language's mapping to weight constraint rules—a form of logic programs. Modeling Tool enables efficient graphical modeling and includes several functions that support long-term management, such as semi-automatic generation of user interfaces. Configuration Tool provides sales configuration functionality for e-commerce. It applies an inference engine based on weight constraint rules (Smodels) to provide consistent and complete inference. Early phases on the track to WeCoTin also affected the development of Smodels.

Evaluation of WeCoTin was multi-faceted. It included the characterization of 26 sales configuration models and run-time performance analysis, both with developed new methods. The method for empirical run-time performance

evaluation simulates a naïve user entering random requirements to a configurator equipped with a real-world configuration model. Performance of the inference engine of WeCoTin (smodels) was on a practical level. We expect that adequate performance would apply to many other products that were suitable for Web-based sales configuration.

WeCoTin has been commercialized.²³ This commercialization constitutes market-based validation of pragmatic relevance “weak market test” of a construct²⁴ (Kasanen, Lukka, & Siitonen, 1993). Kasanen et al. contend that ideas and constructs compete in markets, and an idea’s commercial adoption contributes to its significance. In their view, “even the weak market test is relatively strict—it is probably not often that a tentative construction is able to pass it.”

The modeling language PCML allowed efficient modeling of products for web-based sales configuration. Such mechanisms as inheritance and compositional structure with refinement were useful. The current view of the author is that, with otherwise the same modeling mechanisms (compositional structure, taxonomy including inheritance, attributes, and constraints), it is largely irrelevant what the basic objects are called: “features” as in WeCoTin, “components” as in an earlier version of WeCoTin and most of the literature, “functions” as in conceptualization, or just “objects.”

The author awaits with interest any evidence on whether the SCISDT has an effect on future configurator development, including the use of a weight constraint rule language as the formal basis.

The author holds that recommender-supported configuration systems have significant potential to support the mass customization strategy, reduce mass confusion, and make complex products and services accessible to larger audiences, even in self-service e-commerce scenarios. Thus, an overview of feature value recommendation technologies for configurable offerings was provided. Existing case-based feature value recommendation algorithms were extended by integrating importance weights and similarity metrics. A basic evaluation of the case-based collaborative approach was provided by an empirical study. Users evaluated systems with recommendation support higher with respect to all research hypotheses. Three hypotheses indicating improved choice navigation support were statistically significant, and five other hypotheses showed improvement without statistical significance. Based on this, it seems safe to claim that recommendation technology can, indeed, provide benefits to users. Future work is needed to generalize the algorithms and to integrate recommendation into widely applied configuration systems.

The offered variation of configurable services and the applicability of configurators designed for physical products to configuring services were analyzed in three industries. The offered variation contained *what*-variation defining the desired service outcome, some temporal aspects of *when*-variation, and limited *with-what*-, *how*-, and *where*-variation. The properties of a customer and

²³ Variantum Oy, <http://www.variantum.com/joomla/en/products/varisales-sales-configurator>

²⁴ ‘Construct’ of (Kasanen, Lukka, & Siitonen, 1993) is any subtype of ‘artifact’ of (Hevner et al., 2004) and terminology of this thesis.

other stakeholders (*to whom*), related equipment, and the environment influenced service availability and pricing. Limited process configuration was also required. The required scope of modeling extended beyond the service product itself to ensure the correctness of the configuration and pricing. A conceptual mismatch arises in application of traditional configuration modeling concepts such as component or feature types. Despite these challenges, traditional configurators can manage sales configuration of some, and probably most, configurable service offerings. Extended configurator support may be required to manage several non-commensurate prices (e.g., initiation, periodic, and pay-per-use price elements) and reconfiguration; the service contents need to be adjusted when customer needs, equipment, environment, or other relevant aspects change.

This work contributed toward practical support for configuration of physical and service offerings. The main contributions include a conceptualization for modeling the offered variation of configurable offerings, a novel sales configurator instantiation based on weight constraint rules and a corresponding information systems design theory (SCISDT), understanding about the relationship of services and configuration, and ways to determine personalized feature value recommendations to help users in choice navigation.

References

- 3DSEnovia. (2012). *ENOVIA Variant Configuration Central*. Vélizy-Villacoublay Cedex, France: Dassault Systèmes.
- Adomavicius, G., & Tuzhilin, A. (2005). Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *Knowledge and Data Engineering, IEEE Transactions On*, 17(6), 734-749. doi:10.1109/TKDE.2005.99
- Aldanondo, M., Hadj-Hamou, K., Moynard, G., & Lamothe, J. (2003). Mass customization and configuration: Requirement analysis and constraint based modeling propositions. *Integrated Computer-Aided Engineering*, 10(2), 177-189.
- Aldanondo, M., Rouge, S., & Véron, M. (2000). Expert configurator for concurrent engineering: Cameleon software and model. *Journal of Intelligent Manufacturing*, 11(2), 127-134.
- Aldanondo, M., Véron, M., & Fargier, H. (1999). Configuration in manufacturing industry requirements, problems and definitions. *IEEE SMC'99 Conference Proceedings. 1999 IEEE International Conference on Systems, Man, and Cybernetics*, Tokyo. Vol. 6 1009-1014. doi:10.1109/ICSMC.1999.816691
- Anderson, A. (2005). *Towards tool-supported configuration of services*. (M. Sc., Helsinki University of Technology, Department of Computer Science and Engineering).
- Anderson, A., & Pasanen, M. (2003). *WeCoTin Requirements and architecture (unpublished)*. Espoo, Finland: Helsinki University of Technology, Software Business and Engineering Institute.
- Ardissono, L., Felfernig, A., Friedrich, G., Goy, A., Jannach, D., Meyer, M. H., Petrone, G., Schäfer, R., & Zanker, M. (2002). Personalizing on-line configuration of products and services. *15th European Conference on Artificial Intelligence (ECAI-2001)*, Lyon, France. 225-229.
- Ardissono, L., Felfernig, A., Friedrich, G., Goy, A., Jannach, D., Petrone, G., Schäfer, R., & Zanker, M. (2003). A framework for the development of personalized, distributed web-based configuration systems. *AI Magazine*, 24(3), 93-110. doi:10.1609/aimag.v24i3.1721
- Artale, A., Franconi, E., Guarino, N., & Pazzi, L. (1996). Part-whole relations in object-centered systems: An overview. *Data & Knowledge Engineering*, 20(3), 347-383. doi:10.1016/S0169-023X(96)00013-4
- Asikainen, T., & Männistö, T. (2010). A metamodelling approach to configuration knowledge representation. *International Journal of Mass Customisation*, 3(4), 333-350. doi:10.1504/IJMASSC.2010.037649
- Attardi, G., Cisternino, A., & Simi, M. (1998). Web-based configuration assistants. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing (AI EDAM)*, 12(4), 321-331.
- Axling, T., & Haridi, S. (1996). A tool for developing interactive configuration applications. *The Journal of Logic Programming*, 26(2), 147-168. doi:10.1016/0743-1066(95)00097-6
- Baader, F. (2009). Description logics. In S. Tessaris, E. Franconi, T. Eiter, C. Gutierrez, S. Handschuh, M. Rousset & R. A. Schmidt (Eds.), *Reasoning Web. Semantic Technologies for Information Systems* (pp. 1-39). Berlin, Heidelberg: Springer. doi:10.1007/978-3-642-03754-2
- Baida, Z., Akkermans, H., & Gordijn, J. (2003). Serviguration: towards online configurability of real-world services. *ICEC '03 Proceedings of the 5th international conference on Electronic commerce*, Pittsburgh, Pennsylvania, USA. 111-118. doi:10.1145/948005.948020
- Barker, V. E., O'Connor, D. E., Bachant, J., & Soloway, E. (1989). Expert systems for configuration at Digital: XCON and beyond. *Communications of the ACM*, 32(3), 298-318. doi:10.1145/62065.62067

- Beckett, S. B. (1996). *Achieving competitive advantage through customer value: exploring mass customization as a strategy for service organizations*. (Ph. D., The University of Texas at Arlington).
- Bettman, J. R., Luce, M. F., & Payne, J. W. (1998). Constructive consumer choice processes. *Journal of Consumer Research*, 25(3), 187-217. doi:10.1086/209535
- Blecker, T., Friedrich, G., Kaluza, B., Abdelkafi, N., & Kreutler, G. (2005). *Information and management systems for product customization*. Boston: Springer.
- Böhmman, T., Junginger, M., & Krcmar, H. (2003). Modular service architectures: a concept and method for engineering IT services. *Proceedings of the 36th Annual Hawaii International Conference on System Sciences (HICSS'03)*, Waikoloa, HI, USA. CDROM, 10 pp. doi:10.1109/HICSS.2003.1174189
- Bowen, D. E., & Youngdahl, W. E. (1998). "Lean" service: in defense of a production line approach. *International Journal of Service Industry Management*, 9(3), 207-225. doi:10.1108/09564239810223510
- Bryman, A., & Bell, E. (2007). *Business research methods* (2nd ed.). Oxford, UK: Oxford university press.
- Buckley, F. J. (1993). *Implementing Configuration Management: Hardware, Software and Firmware* IEEE Press Piscataway, NJ, USA.
- Burke, R. (2000). Knowledge-based recommender systems. In A. Kent (Ed.), *Encyclopedia of Library and Information Systems - Volume 69* (pp. 180-201). New York, NY, USA: Marcel Dekker.
- Burke, R. (2002). Hybrid recommender systems: survey and experiments. *User Modeling and User - Adapted Interaction*, 12(4), 331-370. doi:10.1023/A:1021240730564
- Chesbrough, H., & Spohrer, J. (2006). A research manifesto for services science. *Communications of the ACM*, 49(7), 35-40, 33-34 (bibliography). doi:10.1145/1139922.1139945
- Chowdhury, S., & Miles, G. (2006). Customer-induced uncertainty in predicting organizational design: Empirical evidence challenging the service versus manufacturing dichotomy. *Journal of Business Research*, 59(1), 121-129. doi:10.1016/j.jbusres.2005.02.006
- Cook, D. P., Goh, C., & Chung, C. H. (1999). Service typologies: a state of the art survey. *Production and Operations Management*, 8(3), 318-338. doi:10.1111/j.1937-5956.1999.tb00311.x
- Cöster, R., Gustavsson, A., Olsson, T., & Rudström, Å. (2002). Enhancing web-based configuration with recommendations and cluster-based help. *AH'02 Workshop on Recommendation and Personalized in e-Commerce*, Málaga, Spain. 30-40.
- Cunis, R., Günter, A., & Strecker, H. (Eds.). (1991). *The PLAKON-Book* [German: Das PLAKON-Buch: Ein Expertensystemkern für Planungs-und Konfigurierungsaufgaben in technischen Domänen]. London, UK: Springer-Verlag.
- Cunis, R., Günter, A., Syska, I., Peters, H., & Bode, H. (1989). PLAKON - an approach to domain-independent construction. *Proceedings of the Second International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems (IEA/AIE - 89)*, Tullahoma, TN, USA. Vol. 2 866-874. doi:10.1145/67312.67359
- Cunningham, P., Bergmann, R., Schmitt, S., Traphöner, R., Breen, S., & Smyth, B. (2001). Websell: Intelligent sales assistants for the world wide web. *Künstliche Intelligenz*, 15(1), 28-32.
- cyLEDGE. (2013). International Configurator Database, 2013. Retrieved 2/6, 2014, from <http://www.configurator-database.com/services/configurator-database>
- Da Silveira, G., Borenstein, D., & Fogliatto, F. S. (2001). Mass customization: Literature review and research directions. *International Journal of Production Economics*, 72(1), 1-13. doi:10.1016/S0925-5273(00)00079-7
- Damiani, S. R., Brand, T., Sawtelle, M., & Shanzer, H. (2001). *Oracle configurator developer user's guide, release 11i*. Redwood City, CA, USA: Oracle Corporation.
- Darr, T., Klein, M., & McGuinness, D. L. (1998). Special issue: configuration design. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing (AI EDAM)*, 12(4), 293-294.
- Dausch, M., & Hsu, C. (2003). Mass-customize service agreements for heavy industrial equipment. *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, Washington, DC, USA. Vol. 5 4809-4814. doi:10.1109/ICSMC.2003.1245744
- Dausch, M., & Hsu, C. (2006). Engineering service products: the case of mass-customising service agreements for heavy equipment industry. *International Journal of Services Technology and Management*, 7(1), 32-51.

- de Miranda, B., Baida, Z., & Gordijn, J. (2006). Modeling pricing for configuring e-Service bundles. Paper presented at the *19th Bled eCommerce Conference*, Bled, Slovenia. paper 48, 13 pp. Retrieved from [https://domino.fov.uni-mb.si/proceedings.nsf/Proceedings/E09F565B4061153EC125718000315426/\\$File/16_Miranda.pdf](https://domino.fov.uni-mb.si/proceedings.nsf/Proceedings/E09F565B4061153EC125718000315426/$File/16_Miranda.pdf);
- Desisto, R. P. (2004). *Constraints still key for product configurator deployments*. (No. T-22-9419). Stamford, CT, USA: Gartner, Inc.
- Dumas, M., O'Sullivan, J., Heravizadeh, M., Edmond, D., & Ter Hofstede, A. (2002). Towards a Semantic Framework for Service Description. *Semantic Issues in E-Commerce Systems, IFIP TC2 / WG2.6 Ninth Working Conference on Database Semantics*, Hong Kong. 277-291. doi:10.1007/978-0-387-35658-7_17
- Duray, R., Ward, P. T., Milligan, G. W., & Berry, W. L. (2000). Approaches to mass customization: configurations and empirical validation. *Journal of Operations Management*, 18(6), 605-625. doi:10.1016/S0272-6963(00)00043-7
- Dyer, J. S. (2005). MAUT—multiattribute utility theory. In J. Figueira, S. Greco & M. Ehrogett (Eds.), *Multiple criteria decision analysis: state of the art surveys* (pp. 265-292). New York: Springer. doi:10.1007/0-387-23081-5_7
- Edvardsson, B., Gustafsson, A., & Roos, I. (2005). Service portraits in service research: a critical review. *International Journal of Service Industry Management*, 16(1), 107-121. doi:10.1108/09564230510587177
- Edwards, K., Blecker, T., Salvador, F., Hvam, L., & Friedrich, G. E. (Eds.). (2008). *Mass Customization Services. Proceedings of the Joint Conference of the International Mass Customization Meeting 2008 (IMCM'08) and the International Conference on Economic, Technical and Organisational Aspects of Product Configuration Systems (PETO'08)*. Copenhagen, Denmark:
- Falkner, A., Felfernig, A., & Haag, A. (2011). Recommendation technologies for configurable products. *AI Magazine*, 32(3), 99-108. doi:10.1609/aimag.v32i3.2369
- Faltings, B., & Freuder, E. C. (1998). Special Issue on configuration. *IEEE Intelligent Systems*, 13(4), 32-33. doi:10.1109/MIS.1998.708430
- Feldkamp, F., Heinrich, M., & Meyer-Gramann, K. D. (1998). SyDeR—System design for reusability. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing (AI EDAM)*, 12(4), 373-382.
- Felfernig, A., Isak, K., Szabo, K., & Zachar, P. (2007). The VITA financial services sales support environment. *22nd AAAI Conference on Artificial Intelligence (AAAI-07)*, Vancouver, Canada. Vol. 22(2) 1692-1699.
- Felfernig, A. (2007). Standardized configuration knowledge representations as technological foundation for mass customization. *Engineering Management, IEEE Transactions On*, 54(1), 41-56. doi:10.1109/TEM.2006.889066
- Felfernig, A., & Burke, R. (2008). Constraint-based recommender systems: technologies and research issues. *Proceedings of the 10th international conference on Electronic commerce (ICEC '08)*, Innsbruck, Austria. 17-26. doi:10.1145/1409540.1409544
- Felfernig, A., Friedrich, G. E., & Jannach, D. (2000a). Generating product configuration knowledge bases from precise domain extended UML models. *12 th International Conference on Software Engineering and Knowledge Engineering (SEKE 2000)*, Chicago, IL, USA. 284-293.
- Felfernig, A., Friedrich, G., Jannach, D., Stumptner, M., & Zanker, M. (2003). Configuration knowledge representations for Semantic Web applications. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing (AI EDAM)*, 17(1), 31-50. doi:10.1017/S0890060403171041
- Felfernig, A., Friedrich, G., Jannach, D., & Zanker, M. (2002). Configuration knowledge representation using UML/OCL. *UML 2002 — The Unified Modeling Language - Model Engineering, Concepts, and Tools, 5th International Conference, Proceedings (LNCS)*, Dresden, Germany. Vol. 2460 49-62. doi:10.1007/3-540-45800-X_5
- Felfernig, A., Friedrich, G., Schubert, M., Mandl, M., Mairitsch, M., & Teppan, E. (2009). Plausible Repairs for Inconsistent Requirements. *21st International Joint Conference on Artificial Intelligence (IJCAI-09)*, Pasadena, California. 791-796.

- Felfernig, A., Friedrich, G. E., & Jannach, D. (2000b). UML as domain specific language for the construction of knowledge-based configuration systems. *International Journal of Software Engineering and Knowledge Engineering*, 10(4), 449-469. doi:10.1142/S0218194000000249
- Felfernig, A., Mandl, M., Tiihonen, J., & Schubert, M. (2010). Personalized Product Configuration. Paper presented at the *Multikonferenz Wirtschaftsinformatik 2010, 24. PuK-Workshop: Planung/Scheduling und Konfigurieren/Entwerfen*, Göttingen, Germany. 2251-2263. Retrieved from <http://webdoc.sub.gwdg.de/univerlag/2010/mkwi/>
- Felfernig, A., & Schubert, M. (2011). Personalized diagnoses for inconsistent user requirements. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing (AI EDAM)*, 25(2), 175-183. doi:10.1017/S0890060410000612
- Felfernig, A., Stumptner, M., & Tiihonen, J. (2011). Special issue: configuration. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing (AI EDAM)*, 25(2), 113-114. doi:10.1017/S0890060410000569
- Fitzsimmons, J. A., & Fitzsimmons, M. J. (2004). *Service management: operations, strategy, and information technology* (4th ed.). New York, NY, USA: McGraw-Hill/Irwin.
- Fleischanderl, G., Friedrich, G. E., Haselböck, A., Schreiner, H., & Stumptner, M. (1998). Configuring large systems using generative constraint satisfaction. *IEEE Intelligent Systems*, 13(4), 59-68. doi:10.1109/5254.708434
- Forza, C., & Salvador, F. (2002a). Managing for variety in the order acquisition and fulfilment process: The contribution of product configuration systems. *International Journal of Production Economics*, 76(1), 87-98. doi:10.1016/S0925-5273(01)00157-8
- Forza, C., & Salvador, F. (2002b). Product configuration and inter-firm co-ordination: an innovative solution from a small manufacturing enterprise. *Computers in Industry*, 49(1), 37-46.
- Forza, C., & Salvador, F. (2006). *Product information management for mass customization: connecting customer, front-office and back-office for fast and efficient customization*. Hampshire, UK; New York, NY, USA: Palgrave Macmillan.
- Frayman, F., & Mittal, S. (1987). COSSACK: a constraint-based expert system for configuration tasks. In D. Sriram, & R. A. Adey (Eds.), *Knowledge-Based Expert Systems in Engineering: Planning and Design* (pp. 143-166). Woburn, MA, USA: Computational Mechanics Publications.
- Friedrich, G. E., Ryabokon, A., Haselböck, A., Schenner, G., & Schreiner, H. (2011). (Re)configuration based on model generation. Paper presented at the *Second Workshop on Logics for Component Configuration (LoCoCo 2011)*, vol 65 of EPTCS, Perugia, Italy. Vol. 65 of *Electronic Proceedings in Theoretical Computer Science (EPTCS)* 26-35. doi:10.4204/EPTCS.65.3
- Geneste, L., & Ruet, M. (2001). Experience based configuration. *17th International Joint Conference on Artificial Intelligence (IJCAI-2001), Configuration Workshop*, Seattle, WA, USA. 45-49.
- Gilmore, J. H., & Pine, B. J. 2. (1997). The four faces of mass customization. *Harvard Business Review*, 75(1), 91-101.
- Goy, A., & Magro, D. (2004a). Dynamic configuration of a personalized tourist agenda. *Proceedings of the IADIS International Conference WWW/Internet 2004 (ICWI 2004)*, Madrid, Spain. 619-626.
- Goy, A., & Magro, D. (2004b). STAR: a Smart Tourist Agenda Recommender. *16th European Conference on Artificial Intelligence (ECAI-2004), Configuration Workshop*, Valencia, Spain. 8-1-8-8.
- Gregor, S. (2006). The nature of theory in information systems. *MIS Quarterly*, 30(3), 611-642.
- Gregor, S., & Jones, D. (2007). The anatomy of a design theory. *Journal of the Association for Information Systems*, 8(5), 312-335.
- Grönroos, C. (2007). *Service management and marketing: customer management in service competition* (3rd ed.). Chichester, England: John Wiley & Sons.
- Gruber, T. R. (1993). A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2), 199-220. doi:10.1006/knac.1993.1008
- Günter, A., & Hotz, L. (1999). KONWERK - A Domain Independent Configuration Tool. *Configuration Papers from the AAAI Workshop, 1999. AAAI Technical Report WS-99-05*, 125-126.
- Haag, A. (2008). What Makes Product Configuration Viable in a Business? *18th European Conference on Artificial Intelligence (ECAI 2008), Configuration Workshop*, Patras, Greece. 53-54.

- Haag, A., Junker, U., & O'Sullivan, B. (2006). A Survey of Explanation Techniques for Configurators. *17th European Conference on Artificial Intelligence (ECAI 2006), Configuration Workshop*, Riva del Garda, Italy. 8-13.
- Haag, A. (1998). Sales configuration in business processes. *IEEE Intelligent Systems*, 13(4), 78-85. doi:10.1109/5254.708436
- Haag, A., Junker, U., & O'Sullivan, B. (2007). Explanation in product configuration. *IEEE Intelligent Systems*, 22(1), 83-85.
- Hales, H. L. (1992). Automating and integrating the sales function: how to profit from complexity and customization. *Enterprise Integration Strategies*, 9(11), 1-9.
- Hansen, C., & Andreassen, M. M. (2002). Two approaches to synthesis based on the domain theory. In A. Chakrabarti (Ed.), (pp. 93-108). London: Springer. doi:10.1007/978-1-4471-3717-7_6
- Hart, C. W. L. (1995). Mass customization: conceptual underpinnings, opportunities and limits. *International Journal of Service Industry Management*, 6(2), 36-45. doi:10.1108/09564239510084932
- Harvey, J., Lefebvre, L. A., & Lefebvre, E. (1997). Flexibility and technology in services: a conceptual model. *International Journal Operations & Production Management*, 17(1), 29-45. doi:10.1108/01443579710157970
- Häubl, G., & Murray, K. B. (2003). Preference construction and persistence in digital marketplaces: The role of electronic recommendation agents. *Journal of Consumer Psychology*, 13(1-2), 75-91. doi:10.2139/ssrn.964192
- Heatley, J., Agarwal, R., & Tanniru, M. (1995). An evaluation of an innovative information technology — the case of Carrier EXPERT. *Journal of Strategic Information Systems*, 4(3), 255-277. doi:10.1016/0963-8687(95)96805-1
- Hedin, G., Ohlsson, L., & McKenna, J. (1998). Product configuration using object oriented grammars. *System Configuration Management: ECOOP'98 SCM-8 Symposium*. Brussels, Belgium. Vol. LNCS 1439 107-126. doi:10.1007/BFb0053882
- Heinrich, M., & Jüngst, E. W. (1991). A resource-based paradigm for the configuring of technical systems from modular components. *Seventh IEEE Conference on Artificial Intelligence Applications (CAIA-91)*, Miami Beach, FL, USA. Vol. i 257-264. doi:10.1109/CAIA.1991.120878
- Heinrich, M., & Jüngst, E. W. (1996). Configuring Technical Systems from Modular Components. *Configuration—Papers from the 1996 AAAI Fall Symposium. Technical Report FS-96-03*. 111-118.
- Heiskala, M. (2005). *A conceptual model for modeling configurable services from a customer perspective*. (M.Sc. (Eng.), Helsinki University of Technology, Department of Electrical and Communications Engineering).
- Heiskala, M., Anderson, A., Huhtinen, V., Tiihonen, J., & Martio, A. (2003). A Tool for Comparing Configurable Products. *18th International Joint Conference on Artificial Intelligence (IJCAI '03), Workshop on Configuration*, Acapulco, Mexico. 64-69.
- Heiskala, M., Paloheimo, K., & Tiihonen, J. (2005). Mass Customisation of Services: Benefits and Challenges of Configurable Services. *Frontiers of e-Business Research (FeBR) 2005, Conference proceedings of eBRF*, Tampere, Finland. 206-221.
- Heiskala, M., Paloheimo, K., & Tiihonen, J. (2007). Mass customization with configurable products and configurators: a review of benefits and challenges. In T. Blecker, & G. Friedrich (Eds.), *Mass customization information systems in business* (1st ed., pp. 1-32). Hershey, PA, USA & London, UK: IGI Global. doi:10.4018/978-1-59904-039-4.ch001
- Heiskala, M., Tiihonen, J., Anderson, A., & Soininen, T. (2006). Four-Worlds Model for Configurable Services. *Customer Interaction and Customer Integration. Proceedings of the Joint Conference IMCM'06 & PE-TO'06*, Hamburg, Germany. 199-216.
- Heiskala, M., Tiihonen, J., & Soininen, T. (2005). A Conceptual Model for Configurable Services. *19th International Joint Conference on Artificial Intelligence (IJCAI-05), Configuration Workshop*, Edinburgh, Scotland, UK. 19-24.
- Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design science in information systems research. *MIS Quarterly*, 28(1), 75-105.

- Hotz, L., & Günter, A. (2014). Konwerk. In A. Felfernig, L. Hotz, C. Bagley & J. Tiihonen (Eds.), *Knowledge-based Configuration - From Research to Business Cases* (1st ed., pp. 281-295). Waltham, MA, USA: Morgan Kaufmann Publishers.
- Huffman, C., & Kahn, B. E. (1998). Variety for sale: Mass customization or mass confusion? *Journal of Retailing*, 74(4), 491-513. doi:10.1016/S0022-4359(99)80105-5
- Hvam, L., Riis, J., & Hansen, B. L. (2003). CRC cards for product modelling. *Computers in Industry*, 50(1), 57-70. doi:10.1016/S0166-3615(02)00143-4
- Hvam, L., Mortensen, N. H., & Riis, J. (2008). *Product customization* (1st ed.). New York: Springer.
- ISR. (2013). Editorial Statement of Information Systems Research (ISR) journal. Retrieved 2/25, 2014, from <http://pubsonline.informs.org/page/isre/editorial-statement>
- John, U., & Geske, U. (1999). Reconfiguration of technical products using ConBaCon. *Sixteenth National Conference on Artificial Intelligence (AAAI-99), Workshop on Configuration*, Orlando, Florida, USA. 48-53.
- Junker, U., & Mailharro, D. (2003a). The logic of ILOG (J)configurator: Combining constraint programming with a description logic. *18th International Joint Conference on Artificial Intelligence (IJCAI-03), Configuration Workshop*, Acapulco, Mexico. 13-20.
- Junker, U., & Mailharro, D. (2003b). Preference programming: Advanced problem solving for configuration. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing (AI EDAM)*, 17(1), 13-29. doi:10.1017/S089006040317103X
- Kasanen, E., Lukka, K., & Siitonen, A. (1993). The Constructive Approach in Management Accounting Research. *Journal of Management Accounting Research*, 5(1), 243-264.
- Konstan, J. A., Miller, B. N., Maltz, D., Herlocker, J. L., Gordon, L. R., & Riedl, J. (1997). GroupLens: applying collaborative filtering to Usenet news. *Communications of the ACM*, 40(3), 77-87. doi:10.1145/245108.245126
- Kotler, P. (1988). Managing Services. *Marketing management: analysis, planning, implementation, and control* (6th ed., pp. 476-493). Englewood Cliffs, NJ, USA: Prentice-Hall.
- Levitt, T. (1981). Marketing intangible products and product intangibles. *Harvard Business Review*, 59, 94-102.
- Linden, G., Smith, B., & York, J. (2003). Amazon.com recommendations: Item-to-item collaborative filtering. *Internet Computing, IEEE*, 7(1), 76-80. doi:10.1109/MIC.2003.1167344
- Lovelock, C. H. (1983). Classifying services to gain strategic marketing insights. *Journal of Marketing*, 47(3), 9-20. doi:10.2307/1251193
- Lovelock, C., & Gummesson, E. (2004). Whither services marketing? In search of a new paradigm and fresh perspectives. *Journal of Service Research*, 7(1), 20-41. doi:10.1177/1094670504266131
- Ma, Q., Tseng, M. M., & Yen, B. (2002). A generic model and design representation technique of service products. *Technovation*, 22(1), 15-39. doi:10.1016/S0166-4972(00)00085-7
- Mackworth, A. K. (1977). Consistency in networks of relations. *Artificial Intelligence*, 8(1), 99-118. doi:10.1016/0004-3702(77)90007-8
- Mackworth, A. K., & Freuder, E. C. (1985). The complexity of some polynomial network consistency algorithms for constraint satisfaction problems. *Artificial Intelligence*, 25(1), 65-74. doi:10.1016/0004-3702(85)90041-4
- Mailharro, D. (1998). A classification and constraint-based framework for configuration. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing (AI EDAM)*, 12(4), 383-397.
- Männistö, T. (2000). *A conceptual modelling approach to product families and their evolution*. (Doctor of Science (Eng.), Helsinki University of Technology). *Acta Polytechnical Scandinavica, Mathematics and Computing Series, Ma 106*
- McDermott, J. (1982). R1: A Rule-based configurator of computer systems. *Artificial Intelligence*, 19(1), 39-88. doi:10.1016/0004-3702(82)90021-2
- McDermott, J. (1993). R1 ("XCON") at age 12: lessons from an elementary school achiever. *Artificial Intelligence*, 59(1-2), 241-247. doi:10.1016/0004-3702(93)90192-E

- McGuinness, D. L., & Wright, J. R. (1998a). Conceptual modelling for configuration: A description logic-based approach. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing (AI EDAM)*, 12(4), 333-344.
- McGuinness, D. L., & Wright, J. R. (1998b). An industrial-strength description logic-based configurator platform. *IEEE Intelligent Systems*, 13(4), 69-77. doi:10.1109/5254.708435
- McLaughlin, C. P. (1996). Why variation reduction is not everything: a new paradigm for service operations. *International Journal of Service Industry Management*, 7(3), 17-30. doi:10.1108/09564239610122938
- McSherry, D. (2003). Similarity and compromise. *5th International Conference on Case-Based Reasoning (ICCBR-03)*, Trondheim, Norway. Vol. LNCS 2689 291-305. doi:10.1007/3-540-45006-8_24
- Meier, H., & Massberg, W. (2004). Life cycle-based service design for innovative business models. *CIRP Annals-Manufacturing Technology*, 53(1), 393-396. doi:10.1016/S0007-8506(07)60724-0
- Meyer, M. H., & DeTore, A. (2001). PERSPECTIVE: Creating a platform-based approach for developing new services. *The Journal of Product Innovation Management*, 18(3), 188-204. doi:10.1111/1540-5885.1830188
- Mittal, S., & Falkenhainer, B. (1990). Dynamic constraint satisfaction problems. *Proceedings of the Eighth National Conference on Artificial Intelligence (AAAI-90)*, Boston, MA, USA. 25-32.
- Mittal, S., & Frayman, F. (1989). Towards a generic model of configuration tasks. *11th International Joint Conference on Artificial Intelligence (IJCAI-89)*, Detroit, Michigan, USA. Vol. 2 1395-1401.
- Najmann, O., & Stein, B. (1992). A Theoretical Framework for Configuration. *Industrial and Engineering Applications of Artificial Intelligence and Expert Systems: 5th International Conference (IEA/AIE-92)*, Paderborn, Germany. Vol. LNCS 604 441-450. doi:10.1007/BFb0024996
- Nielsen, J. (1993). *Usability Engineering* (1st ed.). Boston: Academic Press.
- Niemi, J. (2007). *A study on the modeling of configurable products and services*. (Unpublished M.Sc. (Eng.)). Tampere University of Technology, Mechanical Engineering Department, Tampere, Finland.
- Nurmilaakso, J. (2004). *WeCotin.calc documentation*. Espoo, Finland: unpublished project documentation of Helsinki University of Technology, Department of Computer Science and Engineering.
- OMG. (2011). *OMG Unified Modeling Language (OMG UML), Infrastructure, version 2.4.1* Object Management Group, Inc. Retrieved from <http://www.omg.org/spec/UML/2.4.1/Infrastructure>
- Oracle. (2004). *Oracle configurator 11i data sheet*. Redwood City, CA, USA: Oracle Corporation;.
- Oracle. (2005). *Peoplesoft Enterprise Configurator - Oracle Data Sheet*. Redwood City, CA, USA: Oracle Corporation.
- Oracle. (2006). *Oracle Configurator R12 - Oracle data sheet* Oracle Corporation. Retrieved from <http://www.expobadge.com/dldev/dc/DKLLoader1.cfm?dcid=2032&type=pdf&aid=ace3f4dd-2baa-4ba9-ad6c-099ce814bfba&caller=as1&shownumber=8027;>
- Oracle. (2007). *Siebel Product & Catalog Management - Oracle data sheet*. Redwood City, CA, USA: Oracle Corporation. Retrieved from <http://www.oracle.com/us/products/applications/siebel/036241.pdf>
- Oracle. (2009). *Oracle Configurator - Oracle data sheet*. Redwood City, CA, USA: Oracle Corporation. Retrieved from <http://www.oracle.com/us/products/applications/046986.pdf>
- Oracle. (2012). *JD Edwards EnterpriseOne configurator - Oracle data sheet*. Redwood City, CA, USA: Oracle Corporation. Retrieved from <http://www.oracle.com/us/media/057429.pdf>
- Papazoglou, M. P., & Georgakopoulos, D. (2003). Service-Oriented Computing. *Communications of the ACM*, 46(10), 25-28. doi:10.1145/944217.944233
- Pasanen, M. (2003). *Warnings and pre-selection packages in a weight constraint rule based configurator*. (Unpublished M.Sc (Eng.)). Helsinki University of Technology, Department of Computer Science and Engineering, Espoo, Finland.
- Pazzani, M. J. (1999). A framework for collaborative, content-based and demographic filtering. *The Artificial Intelligence Review*, 13(5-6), 393-408. doi:10.1023/A:1006544522159
- Peppers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2007). A design science research methodology for information systems research. *Journal of Management Information Systems*, 24(3), 45-77. doi:10.2753/MIS0742-1222240302

- Peltonen, H., Tiihonen, J., & Anderson, A. (2001). *Configurator tool concepts and model definition language*. Espoo, Finland: Working document of Helsinki University of Technology, Software Business and Engineering Institute, Product Data Management Group.
- Pine, B. J. (1993). *Mass customization: the new frontier in business competition* (1st ed.). Boston, MA, USA: Harvard Business School Press.
- PTC. (2012). *Windchill: Managing the complete product lifecycle – from concept to service*. Needham, MA, USA: Parametric Technology Corporation (PTC).
- Resnick, P., & Varian, H. R. (1997). Recommender systems. *Communications of the ACM*, 40(3), 56-58. doi:10.1145/245108.245121
- Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., & Lorensen, W. (1991). *Object-oriented modeling and design* (1st ed.). Englewood Cliffs, NJ, USA: Prentice-Hall.
- Rumbaugh, J., Jacobson, I., & Booch, G. (1999). *The unified modeling language reference manual* (1st ed.). Reading, MA, USA: Addison-Wesley.
- Sabin, D., & Weigel, R. (1998). Product configuration frameworks — a survey. *IEEE Intelligent Systems*, 13(4), 42-49. doi:10.1109/5254.708432
- Salvador, F., de Holan, P. M., & Piller, F. T. (2009). Cracking the code of mass customization. *MIT Sloan Management Review*, 50(3), 71-78.
- Salvador, F., & Forza, C. (2007). Principles for efficient and effective sales configuration design. *International Journal of Mass Customisation*, 2(1-2), 114-127.
- SAP. (2001). *Solution brief - customer relationship management with mySAP telecommunications* [SAP Document ID: 50 048 401 (01/06/10)]. Walldorf, Germany: SAP AG.
- SAP. (2005). *SAP solution brief - dealer management for the telecommunications industry* [SAP Document ID 50 065 332 (05/05)]. Walldorf, Germany: SAP AG.
- Sawtelle, M. (2010). *Oracle Configurator: Fusion Configurator Engine Guide, Release 12.1*. Redwood City, CA, USA: Oracle Corporation.
- Schenner, G., Falkner, A., Ryabokon, A., & Friedrich, G. E. (2013). Solving Object-oriented Configuration Scenarios with ASP. Paper presented at the *15th International Configuration Workshop*, Vienna, Austria. 55-62. Retrieved from <http://ws-config-2013.mines-albi.fr/CWS-2013-Proceedings-Color.pdf>
- Searls, D. B., & Norton, L. M. (1990). Logic-based configuration with a semantic network. *The Journal of Logic Programming*, 8(1-2), 53-73. doi:10.1016/0743-1066(90)90051-6
- Shadish, W. R., Cook, T. D., & Campbell, D. T. (2002). *Experimental and quasi-experimental designs for generalized causal inference* (2nd ed.). Boston: Houghton Mifflin.
- Sharma, N., & Colomb, R. (1998). Mechanising shared configuration and diagnosis theories through constraint logic programming. *The Journal of Logic Programming*, 37(1-3), 255-283. doi:10.1016/S0743-1066(98)10010-9
- Siemens. (2011). *Teamcenter product variation and derivative management* Siemens Product Lifecycle Management Software Inc.
- Silvestro, R., Fitzgerald, L., Johnston, R., & Voss, C. (1992). Towards a classification of service processes. *International Journal of Service Industry Management*, 3(3), 62-75. doi:10.1108/09564239210015175
- Simon, H. A. (1996). *The sciences of the artificial* (3rd ed.). Cambridge, MA, USA: MIT press.
- Simons, P., Niemelä, I., & Soininen, T. (2002). Extending and implementing the stable model semantics. *Artificial Intelligence*, 138(1-2), 181-234. doi:10.1016/S0004-3702(02)00187-X
- Sinz, C., Kaiser, A., & Kuchlin, W. (2003). Formal methods for the validation of automotive product configuration data. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing (AI EDAM)*, 17(1), 75-97. doi:10.1017/S0890060403171065
- Sinz, C., Haag, A., Narodytska, N., Walsh, T., Gelle, E., Sabin, M., Junker, U., O'Sullivan, B., Rabiser, R., Dhungana, D., Grunbacher, P., Lehner, K., Federspiel, C., & Naus, D. (2007). Configuration. *IEEE Intelligent Systems*, 22(1), 78-90. doi:10.1109/MIS.2007.6
- Slater, P. J. P. (1999). PCONFIG: a Web-based configuration tool for Configure-To-Order products. *Knowledge-Based Systems*, 12(5-6), 223-230. doi:10.1016/S0950-7051(99)00016-7
- Soininen, T., & Gelle, E. (1999). Dynamic Constraint Satisfaction in Configuration. *AAAI Workshop on Configuration*, AAAI Technical Report WS-99-05, Orlando, Florida. 95-100.

- Soininen, T., Gelle, E., & Niemelä, I. (1999). A Fixpoint Definition of Dynamic Constraint Satisfaction. *Principles and Practice of Constraint Programming – CP'99, 5th International Conference*, Alexandria, VA, USA. Vol. LNCS 1713 419-433. doi:10.1007/978-3-540-48085-3_30
- Soininen, T., & Tiihonen, J. (1995). *Sales configurator in Datex product data management process* [Finnish: Myynnin konfiguraattorin sijoittuminen Datexin tuotetiedonhallintaprosessiin]. Espoo: Report on seminar course Tik-86.161 Special topics in CIM II, Helsinki University of Technology, Laboratory of Information Processing Science.
- Soininen, T. (2000). *An Approach to Knowledge Representation and Reasoning for Product Configuration Tasks*. (Ph.D., Helsinki University of Technology, Department of Computer Science and Engineering). *Acta Polytechnica Scandinavica, Mathematics and Computing Series*, 111
- Soininen, T., Niemelä, I., Tiihonen, J., & Sulonen, R. (2001). Representing configuration knowledge with weight constraint rules. *AAAI Spring Symposium on Answer Set Programming: Towards Efficient and Scalable Knowledge (AAAI Technical Report SS-01-01)*, Stanford University, CA, USA. 195-201.
- Soininen, T., & Stumptner, M. (2003). Special issue: configuration. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing (AI EDAM)*, 17(1), 1-2. doi:10.1017/S0890060403171016
- Stegmann, R., Leckner, T., Koch, M., & Schlichter, J. (2006). Customer support for the web-based configuration of individualised products. *International Journal of Mass Customisation*, 1(2-3), 195-217.
- Stegmann, R., Koch, M., Lacher, M., Leckner, T., & Renneberg, V. (2003). Generating personalized recommendations in a model-based product configurator system. *18th International Joint Conference on Artificial Intelligence (IJCAI-03), Configuration Workshop*, Acapulco, Mexico. 51-55.
- Steinmetz, S. (1996). *Random House compact unabridged dictionary* (2nd ed.). New York, NY, USA: Random House.
- Stolze, M., & Field, S. (2000). Combining configuration and evaluation mechanisms to support the selection of modular insurance products. *8th European Conference on Information Systems, Trends in Information and Communication Systems for the 21st Century (ECIS 2000)*, Vienna, Austria. 858-865.
- Štorga, M., Andreasen, M. M., & Marjanović, D. (2010). The design ontology: foundation for the design knowledge exchange and management. *Journal of Engineering Design*, 21(4), 427-454. doi:10.1080/09544820802322557
- Stumptner, M. (1997). An overview of knowledge-based configuration. *AI Communications*, 10(2), 111-125.
- Stumptner, M., Friedrich, G. E., & Haselböck, A. (1998). Generative constraint-based configuration of large technical systems. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing (AI EDAM)*, 12(4), 307-320.
- Stumptner, M., Haselböck, A., & Friedrich, G. E. (1994). COCOS - a tool for constraint-based, dynamic configuration. *10th IEEE Conference on Artificial Intelligence for Applications (CAIA-94)*, San Antonio, TX, USA. 373-380. doi:10.1109/CAIA.1994.323651
- Stvilia, B. (2007). A model for ontology quality evaluation. *First Monday*, 12(12), Feb 2, 2014. doi:10.5210/fm.v12i12.2043
- Subbarayan, S. (2005). CLib: configuration benchmarks library. Retrieved 5/7, 2010, from <http://www.itu.dk/research/cla/externals/clib>
- Sundbo, J. (1994). Modulization of service production and a thesis of convergence between service and manufacturing organizations. *Scandinavian Journal of Management*, 10(3), 245-266. doi:10.1016/0956-5221(94)90002-7
- Sundbo, J. (2002). The service economy: standardisation or customisation? *The Service Industries Journal*, 22(4), 93-116. doi:10.1080/714005099
- Sviokla, J. J. (1990). An examination of the impact of expert systems on the firm: the case of XCON. *MIS Quarterly*, 14(2), 127-140. doi:10.2307/248770
- Syrjänen, T. (2000). Including Diagnostic Information in Configuration Models. *First International Conference on Computational Logic (CL 2000)*, London, UK. Vol. LNCS 1861 837-851. doi:10.1007/3-540-44957-4_56
- Syrjänen, T. (2002). *Lparse 1.0 user's manual*. Espoo, Finland: Helsinki University of Technology, Laboratory of Theoretical Computer Science. Retrieved from <http://www.tcs.hut.fi/Software/smodels/>

- Talja, T. (2006). *Usability test of WeCoTin configurator* [Finnish: WeCoTin -konfiguraattorin käytettävyydesti]. Espoo, Finland: Unpublished report: T-121-850 Individual Course on Usability (Käytettävyyden yksilöllinen opintopaketti), Helsinki University of Technology, Software Business and Engineering Institute.
- Tiihonen, J. (1994). *Computer-assisted elevator configuration*. (M.Sc (Eng.), Helsinki University of Technology, Department of Computer Science).
- Tiihonen, J. (1999). *National product configuration survey — customer specific adaptation in the Finnish industry*. (Licentiate of technology (Eng.), Helsinki University of Technology, Department of Computer Science, Laboratory of Information Processing Science).
- Tiihonen, J. (2009). Characterization of 26 configuration models. *21st International Joint Conference on Artificial Intelligence (IJCAI-09), Configuration Workshop*, Pasadena, CA, USA. 69-76.
- Tiihonen, J. (2010). Characterization of configuration knowledge bases. *19th European Conference on Artificial Intelligence (ECAI-2010), Workshop on Intelligent Engineering Techniques for Knowledge Bases (IKBET)*, Lissabon, Portugal. 13-20.
- Tiihonen, J., & Anderson, A. (2005). *WeCoTin architecture, setting up and installation (user manual)*. Espoo, Finland: Helsinki University of Technology.
- Tiihonen, J., & Felfernig, A. (2008). Towards Recommending Configurable Offerings. *18th European Conference on Artificial Intelligence (ECAI 2008), Configuration Workshop*, Patras, Greece. 29-34.
- Tiihonen, J., Felfernig, A., & Mandl, M. (2014). Personalized configuration. In A. Felfernig, L. Hotz, C. Bagley & J. Tiihonen (Eds.), *Knowledge-based Configuration - From Research to Business Cases* (1st ed., pp. 167-179). Waltham, MA, USA: Morgan Kaufmann Publishers. doi:10.1016/B978-0-12-415817-7.00013-X
- Tiihonen, J., Felfernig, A., Zanker, M., & Männistö, T. (2010). Special issue: advances in configuration systems: editorial. *International Journal of Mass Customisation*, 3(4), 311-315.
- Tiihonen, J., Heiskala, M., Paloheimo, K., & Anderson, A. (2007). Applying the Configuration Paradigm to Mass-customize Contract Based Services. *Extreme Customization: Proceedings of the MCPC 2007 World Conference on Mass Customization & Personalization*, Massachusetts Institute of Technology, MA, USA. (CDROM, paper ID MCPC-134-2007, section 7.5.3) 26 pp.
- Tiihonen, J., Lehtonen, T., Soininen, T., Puikkinen, A., Sulonen, R., & Riitahuhta, A. (1999). Modeling configurable product families. *12th International Conference on Engineering Design (ICED'99)*, Munich, Germany, 1139-1142.
- Tiihonen, J., Soininen, T., Niemelä, I., & Sulonen, R. (2003). A Practical Tool for Mass-Customising Configurable Products. *Proceedings of the 14th International Conference on Engineering Design*, Stockholm, Sweden. CDROM, paper number 1290, 10 pp.
- Tiihonen, J., Lehtonen, T., Soininen, T., Pulkkinen, A., Sulonen, R., & Riitahuhta, A. (1998). Modeling configurable product families. *4th WDK workshop on product structuring: design of product families*, Delft University of Technology, Netherlands. 29-50.
- Tiihonen, J., & Soininen, T. (1996). State-of-the-practice in product configuration - a survey of 10 cases in the Finnish industry. *Knowledge Intensive CAD*. Espoo, Finland. Vol. 1 of *IFIP Advances in Information and Communication Technology* 95-114.
- Tiihonen, J., & Soininen, T. (1997a). *Product configurators - information system support for configurable products*. (No. TKO-B 137). Espoo: Helsinki University of Technology.
- Tiihonen, J., & Soininen, T. (1997b). *Product Configurators: Information System Support for Configurable Products. Using Information Technology During the Sales Visit* () Hewson Consulting Group.
- Tiihonen, J., Soininen, T., Männistö, T., & Sulonen, R. (1998). Configurable products — lessons learned from the Finnish industry. *Proceedings of the 2nd International Conference on Engineering Design and Automation (ED&A '98)*, Maui, Hawaii, USA. (CDROM, paper no. 368) 6 pp.
- Tiihonen, J., Soininen, T., Niemelä, I., & Sulonen, R. (2002). Empirical testing of a weight constraint rule based configurator. *15th European Conference on Artificial Intelligence (ECAI-2002), Configuration Workshop*, Lyon, France. 17-22.

- Trentin, A., Perin, E., & Forza, C. (2013). Sales configurator capabilities to avoid the product variety paradox: Construct development and validation. *Computers in Industry*, 64(4), 436-447. doi:10.1016/j.compind.2013.02.006
- Vargo, S. L., & Lusch, R. F. (2004a). Evolving to a new dominant logic for marketing. *Journal of Marketing*, 68(1), 1-17. doi:10.1509/jmkg.68.1.1.24036
- Vargo, S. L., & Lusch, R. F. (2004b). The four service marketing myths: remnants of a goods-based, manufacturing model. *Journal of Service Research*, 6(4), 324-335. doi:10.1177/1094670503262946
- Von Winterfeldt, D., & Edwards, W. (1986). *Decision analysis and behavioral research* (1st ed.). Cambridge, UK: Cambridge University Press.
- W3C. (2008). Extensible Markup Language (XML) 1.0. Retrieved 2/25, 2014, from <http://www.w3.org/TR/REC-xml/>
- Wang, Y., & Tseng, M. M. (2011). Adaptive attribute selection for configurator design via Shapley value. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 25(2), 185-195. doi:10.1017/S0890060410000624
- Warmer, J., & Kleppe, A. (2003). *The object constraint language: getting your models ready for MDA* (2nd ed.). Boston, MA, USA: Addison-Wesley.
- Werthner, H., & Ricci, F. (2004). E-commerce and tourism. *Communications of the ACM*, 47(12), 101-105. doi:10.1145/1035134.1035141
- Wielinga, B., & Schreiber, G. (1997). Configuration-design problem solving. *IEEE Expert*, 12(2), 49-56. doi:10.1109/64.585104
- Wikipedia. (2014). XML. Retrieved 2/26, 2014, from <http://en.wikipedia.org/wiki/XML>
- Wilson, D. R., & Martinez, T. R. (1997). Improved heterogeneous distance functions. *Journal of Artificial Intelligence Research*, 6, 1-34.
- Wimmer, A., Mehla, J. I., & Klein, T. (2003). Object Oriented Product Meta-Model for the Financial Services Industry. *2nd Interdisciplinary World Congress on Mass Customization and Personalization (MCPC'03)*, Munich, Germany. (Retrieved from: <http://scg.mit.edu/images/MCPC%202003/site/9-MCP%20Information%20systems/3-Wimmer.pdf>) 9 pp.
- Winter, R. (2001). Mass customization and beyond – evolution of customer centricity in financial services. In C. Rautenstrauch, R. Seelmann-Eggebert & K. Turowski (Eds.), *Moving Into Mass Customization: Information Systems and Management Principles* (pp. 197-213). Berlin Heidelberg: Springer. doi:10.1007/978-3-642-56192-4_12
- Wright, J. R., McGuinness, D. L., Foster, C. H., & Vesonder, G. T. (1995). Conceptual Modeling Using Knowledge Representation: Configurator Applications. *14th International Joint Conference on Artificial Intelligence (IJCAI-95), workshop on Artificial Intelligence in Distributed Information Networks*, Montreal, Quebec, Canada.
- Wright, J. R., Weixelbaum, E. S., Vesonder, G. T., Brown, K. E., Palmer, S. R., Berman, J. I., & Moore, H. H. (1993). A knowledge-based configurator that supports sales, engineering, and manufacturing at AT&T network systems. *AI Magazine*, 14(3), 69-80. doi:10.1609/aimag.v14i3.1055
- Yin, R. K. (2009). *Case Study Research: Design And Methods* (4th ed.). Thousand Oaks, CA, USA: Sage Publications, Inc.
- Yu, B., & Skovgaard, H. J. (1998). A configuration tool to increase product competitiveness. *IEEE Intelligent Systems*, 13(4), 34-41. doi:10.1109/5254.708431
- Zanker, M., Aschinger, M., & Jessenitschnig, M. (2007). Development of a Collaborative and Constraint-Based Web Configuration System for Personalized Bundling of Products and Services. *8th International Conference on Web Information Systems Engineering WISE 2007*, Nancy, France. Vol. LNCS 4831 273-284. doi:10.1007/978-3-540-76993-4
- Zanker, M., Aschinger, M., & Jessenitschnig, M. (2010). Constraint-based personalised configuring of product and service bundles. *International Journal of Mass Customisation*, 3(4), 407-425. doi:10.1504/IJMASSC.2010.037653
- Zeithaml, V. A., Parasuraman, A., & Berry, L. L. (1985). Problems and strategies in services marketing. *Journal of Marketing*, 49(2), 33-46.

Configurable products can be adapted to individual requirements efficiently – a way of mass customization. Design Science artifacts for the sales configuration of physical products and services were developed to advance the state of the art.

An object-oriented conceptualization for configuration knowledge was developed. Based on this, 'WeCoTin', a novel configurator was created. Its high-level, object-oriented, graphical modeling has formal semantics. Web-based sales configuration functionality is provided. Evaluation is based on 26 offerings. Inspired by novelty of WeCoTin, an information systems design theory for sales configurators is proposed.

We identify scenarios where recommendation technologies can support users. Extensions to existing case-based feature value recommendation technologies are proposed and evaluated empirically.

Mass customization of services by configuration is crucial. We discuss the offered variation of configurable services and the applicability of configurators designed for physical products.



ISBN 978-952-60-5894-8
 ISBN 978-952-60-5895-5 (pdf)
 ISSN-L 1799-4934
 ISSN 1799-4934
 ISSN 1799-4942 (pdf)

Aalto University
School of Science
Department of Computer Science and Engineering
www.aalto.fi

**BUSINESS +
 ECONOMY**

**ART +
 DESIGN +
 ARCHITECTURE**

**SCIENCE +
 TECHNOLOGY**

CROSSOVER

**DOCTORAL
 DISSERTATIONS**