# Sales Configurator Information Systems Design Theory

**Juha Tiihonen**[1] and **Tomi Männistö**[2] and **Alexander Felfernig**[3]

**Abstract.** We look for means to advance the field of configuration systems via research that is performed rigorously and methodologically with the aim of theory creation. Specifically, we explore the use of *Information Systems Design Theory* (*ISDT*) as a framework for defining a design science theory for sales configurator construction. ISDT 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". The components of ISDT include purpose and scope, constructs, principles of form and function, artifact mutability, testable propositions, and justificatory knowledge. Generalizing from the novel principles of our earlier work applied in the construction of a sales configuration system called WeCoTin, we present the Sales Configurator Information Systems Design Theory SCISDT. SCISDT aims to support development of generic configurators (aka configuration toolkits) that enable the creation of configurator instantiations for individual companies or product lines to provide choice navigation capability.

## 1 Introduction

Underlying this paper is research that attempted to answer the research question "How to construct a practical and computationally well-founded sales configurator?" [1]. As a part of that research, a generic sales configurator was constructed and evaluated [2]. The configurator was named WeCoTin.

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 [3]. Major research efforts have been devoted to configurators applicable to solving general configuration tasks instead of a specific domain. These include COSSACK [4], PLAKON [5, 6] and its successor KONWERK [7, 8], and COCOS [9]. In addition, a large number of commercial general-purpose configurators exist. Trilogy SalesBUILDER [10] was among the first. ILOG offered a generic configuration engine to be used in other vendors' systems [11, 12]. Anderson [13] identified 30 vendors by their Web pages. In addition, prominent enterprise resource planning systems and CRM vendors have one or more configurators, e.g., SAP [14, 15] and Oracle [16-19].

There exists both numerous individual configurator instantiations and general-purpose configurators that enable the creation of such instantiations. However, developing such artifacts is not a scientific contribution as such and deeper principles are required. Many of the approaches to configurator construction could have been conducted within a Design Science framework but have not necessarily been presented as such. In addition, scientific knowledge on different approaches and means for building configurators has been published in different fields of research. For example, a procedure for implementing configurator instantiations based on generic configurators has been proposed [20] and sound principles and requirements on user interaction of configurators have been presented [21, 22]. However, any theories from the design perspective of generic configurator systems are still nonexistent. This view is supported by an identified need for formal configuration models and inference tools for providing systematic and comprehensive solutions to practitioners [22].

Thus, we see that it is possible to advance the field of configuration systems via research that is performed rigorously and methodologically with the aim of theory creation. Specifically, we explore the use of *Information Systems Design Theory* (*ISDT*) [23] as a framework for defining a design science theory for configurator construction. The underlying idea is that an ISDT can be applied as a prescription when constructing similar artefacts. However, an ISDT must be applied and interpreted in the context of application in an intelligent manner. For example, all aspects of the prescription may not apply in the context or other ISDTs may be applicable as sub-theories.

We use the construction of the WeCoTin sales configurator as a basis for the theory and as an example for illustrating the different parts of the theory.

In the following, we first briefly summarize the existing knowledge and principles behind the creation of configurator systems (Section 2). Thereafter in Section 3, we introduce the Design Science research approach. Section 0 outlines the WeCoTin sales configurator based on our earlier work [1, 2] and introduces ISDT and presents our proposal for the *Sales Configurator Information Systems Design Theory (SCISDT)*. Section 5 concludes.

## 2 Principles of configurators

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.

### 2.1 Configuration knowledge modeling

Configuration knowledge modeling offers ways to represent configuration models, requirements, and configurations. Three primary types of configuration modeling conceptualizations can be

[1] Department of Computer Science and Engineering, Aalto University, Espoo, Finland, email: juha.tiihonen@aalto.fi
[2] Department of Computer Science, Helsinki University, Helsinki, Finland, email: tomi.mannisto@cs.helsinki.fi
[3] Institute for Software Technology, Graz University of Technology, Graz, Austria email: alexander.felfernig@ist.tugraz.at

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*. 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 [24] Sabin and Weigel [25], and Hubaux et al. [26]. Of these methods, constraint satisfaction is the most widely applied. The second type is *configuration-domain-specific conceptualizations*, which are independent of problem-solving methods. These can be roughly classified as *connection-based* [27], *resource-based* [28], *structure-based* [5], or *function-based* [29] approaches. The conceptualizations have little in common, other than the central notion of a component.

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. An example of such a conceptualization is [30]. Unified conceptualizations may include *component types* and their *compositional structure*, *attributes* and *topological concepts* such as *ports* for specifying connectivity. *Resources* 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. *Functions* 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. Concepts discussed above are organized in a taxonomical structure with supertypes, subtypes, and support for inheritance. *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. A similar synthesis as [30] is based on a representation that employs Unified Modeling Language (UML) [31] with specific stereotypes and Object Constraint Language (OCL) [32], was proposed for modeling configuration knowledge [33-37]. The stereotypes include the connection-oriented and resource-oriented concepts along with a taxonomical hierarchy of component types [33-35, 37].

## 2.2 Problem solving

Numerous problem-solving methods have been applied to configuration tasks; several overviews of the topic exist. A recent overview of problem solving in configurators is provided in [38]. In their taxonomy of types of problem-solving methods for design and configuration, Wielinga and Schreiber [39] 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 authors consider 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 [12, 40, 41]. Many authors, e.g., Desisto [42] and Haag, Junker & O'Sullivan [43][2], 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 [45].

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 [46]. Description logics [47] have been applied [48-50]. Constraint logic programming has also been applied [51]. Furthermore, a method has been proposed to translate configuration domain modeling concepts into weight constraint rules [52, 53]. Following this idea, an experimental system, OOASP, showed the feasibility of checking a configuration, completing a configuration, and performing reconfiguration [54].

Sometimes different problem-solving methods have been combined, such as description logic with constraint satisfaction [11].

## 2.3 Other aspects

Principles of configurators include numerous less technical aspects. An overview of configuration systems and current topics is given in [55]. Here, we do not attempt to provide a full treatment of these aspects, and we recognize that there is still significant room for future research. Examples of identified configuration related research challenges include personalized configuration, community-based configuration (by a group of users), standardized configuration knowledge representations, intelligent user interfaces for configuration knowledge acquisition, intelligent testing and debugging, and unobtrusive preference elicitation [56]. To our knowledge, it is not common for generic configuration systems to directly support providing the user support capabilities proposed to avoid the product variety paradox [21]: focused navigation, flexible navigation, easy comparison, benefit-cost communication, and user-friendly product-space description capabilities. Many sales configurators even struggle on aspects like consistency checking [22]. However, the application of configurators in business and corresponding effects (e.g., on organization, processes, business performance), and configurator user interaction aspects are relevant and gaining momentum [21, 22, 57-61]. A number of books guide companies on information management required by mass customization, configurator classifications, and selecting a configurator [20, 59, 62].

# 3 Design Science and theory

The Design Science approach creates and evaluates IT artifacts intended to solve identified organizational problems [63]. The approach is gaining popularity as a framework for research of constructive nature.

[2] An essay in [44] that is based on the Configuration Workshop of the 17th European Conference on Artificial Intelligence (ECAI 2006).

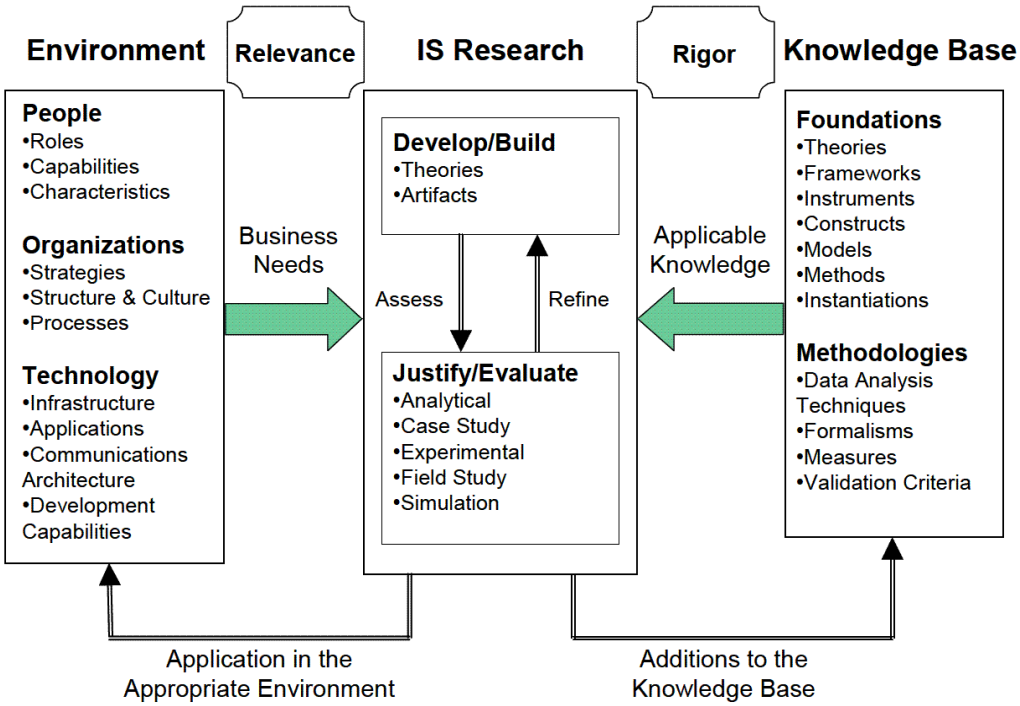


**Figure 1.** Information Systems Research Framework [63], redrawn

Hevner et al. [63] characterize the Design Science approach as follows (see Figure 1). 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.

Gregor [64] discussed the nature of theory in the discipline of Information Systems and presented five theory types (see Table 1). Of these, the most relevant to configuration research, and Design Science in more general, 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). Continuing the idea, Gregor and Jones [23] posit 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. According to Gregor [64], 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 [23] further refine the idea into *elements of information system theory*. They have identified 8 components; see **Table 2**.

**Table 1.** A Taxonomy of Theory Types in Information Systems Research [64](p. 620)

| Theory Type | Distinguishing Attributes |
|---|---|
| I. Analysis | Says what is. The theory does not extend beyond analysis and description. No causal relationships among phenomena are specified and no predictions are made. |
| II. Explanation | Says what is, how, why, when, and where. The theory provides explanations but does not aim to predict with any precision. There are no testable propositions. |
| III Prediction | Says what is and what will be. The theory provides predictions and has testable propositions but does not have well-developed justificatory causal explanations. |
| IV. Explanation and prediction | Says what is, how, why, when, where, and what will be. Provides predictions and has both testable propositions and causal explanations. |
| V Design and action | Says how to do something. The theory gives explicit prescriptions (e.g., methods, techniques, principles of form and function) for constructing an artifact. |

**Table 2** Components of Information Systems Design Theory [23] and Sales Configurator Information Systems Design Theory (SCISDT).

| Component | ISDT component Description [23] | SCISDT component description (as explicated by WeCoTin) |
|---|---|---|
| *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. | A web-based sales configurator that fulfills a set of major requirements |
| 2) Constructs | Representations of the entities of interest in the theory. | Concepts of configuration knowledge [30], product configuration modeling language PCML, weight constraint rule language. |
| 3) Principle of form and function | The abstract "blueprint" or architecture that describes an IS artifact, either product or method / intervention. | A high-level architecture and main functions of components was presented along with main working principles [2, 65, 66] |
| 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. | 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 | Truth statements about the design theory. | 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 underlying knowledge or theory from the natural or social or design sciences that gives a basis and explanation for the design (kernel theories). | 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 components* | | |
| 7) Principles of implementation | A description of processes for implementing the theory (either product or method) in specific contexts. | To be discussed in an extended version of this paper. |
| 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. | WeCoTin. To be discussed in an extended version of this paper. |

# 4 WeCotin and Sales Configurator Information Systems Design Theory

## 4.1 WeCoTin sales configurator

WeCoTin consists of two main components: a graphical modeling environment *Modeling Tool* and a web-based application *WeCoTin Configuration Tool* that supports the configuration task. WeCoTin Configuration Tool enables users to configure products over the web using a standard browser. The user interface for end users is dynamically generated.

WeCoTin Modeling Tool is used for creating and editing configuration models and additional 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 conceptually based on a function-oriented subset of the configuration knowledge conceptualization of Soininen et al. [30].

WeCoTin is computationally well founded because it was constructed based on the idea of translation of configuration knowledge into weight constraint rules [52, 53]. 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 is implemented using the Java 2 Platform and Java programming language, except for the component Inference Engine, which consists of smodels and lparse programs of the Smodels system that are implemented in C++, and user interface components that employ some JavaScript to generate the HTML and CSS-based web-based UI. XML is applied for some user interface definitions, price lists, and calculation definitions.

## 4.2 Purpose and scope

Companies with a mass customization strategy need to provide choice navigation capability [67]. 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. The requirements are summarized in [2, 66] and they include: A (sales) configurator should enable

- easy set-up without programming (excluding integrations),
- fluent modeling of products by product experts based on a well-founded high-level modeling conceptualization,
- easy maintenance of configuration knowledge.

In addition, we wanted to experiment with applying answer set programming for problem solving combined with a higher-level configuration modeling and consistent and complete inference.

## 4.3 Constructs

ISDT *constructs* represent the entities that are of interest in the theory, and corresponding terms should be defined as clearly as possible [23].

In the context of this work, it is somewhat challenging to draw the line between the constructs and principles of form and function. Relevant 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 constructs.

Underlying these as subsystems are the inference engine *Smodels* [68], its modeling language *weight constraint rule language (WCRL),* and the method of translating configuration knowledge to WCRL [53]. These underlying subsystems were developed outside the scope of the WeCoTin construction.

It is noteworthy that the conceptualization was constructed in such a way that 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 variability. Taxonomy with inheritance generalizes the approach. The full conceptualization also supports connection-oriented constructs and resources that have proven to be useful in earlier work. All these can be given formal semantics by mapping them to a formal language.

## 4.4 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" [23].

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 constructs, which match how the product experts think of configuration and yet can be provided with formal semantics and automatically mapped to a form suitable for inference; and
- the end-user interface, creation of which does not require programming, but is, for example, generated utilizing 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 an incomplete or inconsistent configuration. Price is an integral element that must be managed within the scope of a configuration task.

A hierarchy of modeling languages needs to match the layered architecture. In the case of WeCoTin, 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. This principle provides theoretical grounding and allows for sound and complete inference.

We feel that future configurators should support recommendation functionality to support users with choice navigation. Case-based recommendation approaches seem to be potentially viable (e.g. [69]), but further research is required. Future sales configurator ISDTs should address user interaction more thoroughly, e.g. along the lines of [21, 22].

## 4.5 Artifact mutability

WeCoTin has several internal interfaces that enable replacement of major components. For example, Smodels could be relatively easily replaced with another inference engine based on answer set programming. 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).

However, 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.

## 4.6 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.

Created 26 sales configuration models were characterized in terms of size and modeling constructs that were applied [70]. 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 organizations representing machine industry, healthcare, telecommunications services, insurance services, maintenance services, software configuration, and construction. The created models were small, but representative of the Finnish industry. Among larger models was 'Broadband' that had 66 feature types, 453 effective attributes (the sum of inherited and locally defined attributes in concrete types) and 43 type level "generic" constraints. A semi-automatically generated Linux model had 626 feature types, 4369 effective attributes, and 2380 constraints.

WeCoTin had demonstrably adequate performance with the four models that were systematically tested [71]. 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. For the Linux model, achieving sufficient performance would require at least the capability to control when full inference (with finding a configuration) is performed, and possibly other optimizations.

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.

### 4.7 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 [72-75].

PCML allows the variability of products to be expressed on a high level that product experts can understand. Furthermore, the modeling constructs 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.

New methods of characterizing configuration models and measuring configurator performance were developed [70, 71].

Numerous configuration models based on the variability of real offerings were developed [2, 70]. These show how WeCoTin could be applied in respective companies to provide choice navigation support.

## 5 Conclusions

In this paper, we presented, to our knowledge, the first attempt to construct an Information System Design in the context of configuration systems. An ISDT for sales configurators (SCISDT) fulfilling a set of major requirements was presented. SCISDT is based on the design of WeCoTin, a sales configurator that supports mass customization of complex products.

The main components of SCISDT are as follows. The purpose and scope are to construct a web-based sales configurator that fulfills a set of major requirements. The major constructs include a high-level object-oriented configuration modeling language that is based on a well-founded conceptualization that can be mapped to a language with an inference engine to support the configuration task. The principles of form and function include a high-level layered architecture with a matching hierarchy of modeling languages. Artifact mutability includes several internal interfaces and built-in flexibility with respect to numerous aspects that allow for application of the constructed sales configurator more widely than for one specific domain only. The main testable propositions are capability to model and configure real products and adequate performance. Justificatory knowledge includes providing the major modeling constructs clear formal semantics by mapping them to a language with appropriate formal semantics and support for the required inference capabilities.

Although we specifically addressed sales configurators, the Design Science approach can potentially be applied in other configuration related contexts. The authors view that applying the Design Science approach can help to ensure the rigor and relevance of configuration research. Contributions can be the additions to the knowledge base as suggested by Hevner et al. [63], or (ISDT) theories.

## Acknowledgements

We thank DIGILE, TEKES and related companies for financial support; this work has been partially funded by DIGILE SHOK program Need 4 Speed (N4S). We also express our gratitude to companies that have offered us access in the context of earlier research that this work is based on.

## References

[1] J. Tiihonen, *Support for Configuration of Physical Products and Services. Manuscript Submitted to pre-examination for the degree of Doctor of Science (Technology).* Helsinki, Finland: Unigrafia, 2014.

[2] J. Tiihonen, M. Heiskala, A. Anderson and T. Soininen, "WeCoTin–A practical logic-based sales configurator," *AI Communications,* vol. 26 (1), pp. 99-131, 2013.

[3] J. McDermott, "R1: A Rule-based configurer of computer systems," *Artificial Intelligence,* vol. 19 (1), pp. 39-88, 1982.

[4] F. Frayman and S. Mittal, "COSSACK: A constraint-based expert system for configuration tasks," in *Knowledge-Based Expert Systems in Engineering: Planning and Design*, D. Sriram and R. A. Adey, Eds. Woburn, MA, USA: Computational Mechanics Publications, 1987, pp. 143-166.

[5] R. Cunis, A. Günter, I. Syska, H. Peters and H. Bode, "PLAKON - an approach to domain-independent construction," in *Proceedings of the Second International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems (IEA/AIE - 89),* Tullahoma, TN, USA, 1989, pp. 866-874.

[6] R. Cunis, A. Günter and H. Strecker, Eds., *The PLAKON-Book.* London, UK: Springer-Verlag, 1991.

[7] A. Günter and L. Hotz, "KONWERK - A domain independent configuration tool," in *Configuration Papers from the AAAI Workshop, 1999. AAAI Technical Report WS-99-05,* 1999, pp. 125-126.

[8] L. Hotz and A. Günter, "Konwerk," in *Knowledge-Based Configuration - from Research to Business Cases*, 1st ed., A. Felfernig, L. Hotz, C. Bagley and J. Tiihonen, Eds. Waltham, MA, USA: Morgan Kaufmann Publishers, 2014, pp. 281-295.

[9] M. Stumptner, A. Haselböck and G. E. Friedrich, "COCOS - a tool for constraint-based, dynamic configuration," in *10th IEEE Conference on Artificial Intelligence for Applications (CAIA-94),* San Antonio, TX, USA, 1994, pp. 373-380.

[10] H. L. Hales, "Automating and integrating the sales function: how to profit from complexity and customization," *Enterprise Integration Strategies,* vol. 9 (11), pp. 1-9, 1992.

[11] U. Junker and D. Mailharro, "The logic of ILOG (J)configurator: Combining constraint programming with a description logic," in *18th International Joint Conference on Artificial Intelligence (IJCAI-03), Configuration Workshop,* Acapulco, Mexico, 2003, pp. 13-20.

[12] D. Mailharro, "A classification and constraint-based framework for configuration," *Artificial Intelligence for Engineering Design, Analysis and Manufacturing (AI EDAM),* vol. 12 (4), pp. 383-397, 1998.

[13] A. Anderson, *Towards Tool-Supported Configuration of Services.* M. Sc. thesis, Espoo: Helsinki University of Technology, Department of Computer Science and Engineering, 2005.

[14] A. Haag, "Sales configuration in business processes," *IEEE Intelligent Systems,* vol. 13 (4), pp. 78-85, 1998.

[15] A. Haag, ""Dealing" with configurable products in the SAP business suite," in *19th International Joint Conference on Artificial Intelligence (IJCAI-05), Configuration Workshop,* Edinburgh, Scotland, UK, 2005, pp. 68-71.

[16] S. R. Damiani, T. Brand, M. Sawtelle and H. Shanzer, "Oracle configurator developer user's guide, release 11i," 2001.

[17] M. Sawtelle, "Oracle Configurator: Fusion Configurator Engine Guide, Release 12.1," 2010.

[18] Oracle, "Peoplesoft Enterprise Configurator - Oracle Data Sheet," vol. 2009, 2005.
[19] Oracle. Siebel product & catalog management - oracle data sheet. 2007. Available: http://www.oracle.com/us/products/applications/siebel/036241.pdf.
[20] L. Hvam, N. H. Mortensen and J. Riis, *Product Customization.* New York: Springer, 2008.
[21] A. Trentin, E. Perin and C. Forza, "Sales configurator capabilities to avoid the product variety paradox: Construct development and validation," *Comput. Ind.,* vol. 64 (4), pp. 436-447, 2013.
[22] E. K. Abbasi, A. Hubaux, M. Acher, Q. Boucher and P. Heymans, "The anatomy of a sales configurator: An empirical study of 111 cases," in *Advanced Information Systems Engineering - 25th International Conference, CAiSE 2013,* Valencia, Spain, 2013, pp. 162-177.
[23] S. Gregor and D. Jones, "The anatomy of a design theory," *Journal of the Association for Information Systems,* vol. 8 (5), pp. 312-335, 2007.
[24] M. Stumptner, "An overview of knowledge-based configuration," *AI Communications,* vol. 10 (2), pp. 111-125, 1997.
[25] D. Sabin and R. Weigel, "Product configuration frameworks — a survey," *IEEE Intelligent Systems,* vol. 13 (4), pp. 42-49, 1998.
[26] A. Hubaux, D. Jannach, C. Drescher, L. Murta, T. Männistö, K. Czarnecki, P. Heymans, T. Nguyen and M. Zanker, "Unifying software and product configuration: A research roadmap," in *Proceedings of the Workshop on Configuration at ECAI 2012 (ConfWS'12),* Montpellier, France, 2012, pp. 31-35.
[27] S. Mittal and F. Frayman, "Towards a generic model of configuration tasks," in *11th International Joint Conference on Artificial Intelligence (IJCAI-89),* Detroit, Michigan, USA, 1989, pp. 1395-1401.
[28] M. Heinrich and E. W. Jüngst, "A resource-based paradigm for the configuring of technical systems from modular components," in *Seventh IEEE Conference on Artificial Intelligence Applications (CAIA-91),* Miami Beach, FL, USA, 1991, pp. 257-264.
[29] O. Najmann and B. Stein, "A theoretical framework for configuration," in *Industrial and Engineering Applications of Artificial Intelligence and Expert Systems: 5th International Conference (IEA/AIE-92),* Paderborn, Germany, 1992, pp. 441-450.
[30] T. Soininen, J. Tiihonen, T. Männistö and R. Sulonen, "Towards a general ontology of configuration," *AI EDAM,* vol. 12 (4), pp. 357-372, 1998.
[31] J. Rumbaugh, I. Jacobson and G. Booch, *The Unified Modeling Language Reference Manual.* Reading, MA, USA: Addison-Wesley, 1999.
[32] J. Warmer and A. Kleppe, *The Object Constraint Language: Getting Your Models Ready for MDA.* Boston, MA, USA: Addison-Wesley, 2003.
[33] A. Felfernig, G. E. Friedrich and D. Jannach, "Generating product configuration knowledge bases from precise domain extended UML models," in *12 Th International Conference on Software Engineering and Knowledge Engineering (SEKE 2000),* Chicago, IL, USA, 2000, pp. 284-293.
[34] A. Felfernig, G. E. Friedrich and D. Jannach, "UML as domain specific language for the construction of knowledge-based configuration systems," *International Journal of Software Engineering and Knowledge Engineering,* vol. 10 (4), pp. 449-469, 2000.
[35] A. Felfernig, G. Friedrich, D. Jannach and M. Zanker, "Configuration knowledge representation using UML/OCL," in *UML 2002 — the Unified Modeling Language - Model Engineering, Concepts, and Tools, 5th International Conference, Proceedings (LNCS),* Dresden, Germany, 2002, pp. 49-62.
[36] A. Felfernig, G. Friedrich, D. Jannach, M. Stumptner and M. Zanker, "Configuration knowledge representations for Semantic Web applications," *Artificial Intelligence for Engineering Design, Analysis and Manufacturing (AI EDAM),* vol. 17 (1), pp. 31-50, 2003.
[37] A. Felfernig, "Standardized configuration knowledge representations as technological foundation for mass customization," *Engineering Management, IEEE Transactions On,* vol. 54 (1), pp. 41-56, 2007.
[38] L. Hotz, A. Felfernig, M. Stumptner, A. Ryabokon, C. Bagley and K. Wolter, "Configuration knowledge representation and reasoning," in *Knowledge-Based Configuration - from Research to Business Cases*, A. Felfernig, L. Hotz, C. Bagley and J. Tiihonen, Eds. Waltham, MA, USA: Morgan Kaufmann, 2014, pp. 41-72.
[39] B. Wielinga and G. Schreiber, "Configuration-design problem solving," *IEEE Expert,* vol. 12 (2), pp. 49-56, 1997.
[40] S. Mittal and B. Falkenhainer, "Dynamic constraint satisfaction problems," in *Proceedings of the Eighth National Conference on Artificial Intelligence (AAAI-90),* Boston, MA, USA, 1990, pp. 25-32.
[41] G. Fleischanderl, G. E. Friedrich, A. Haselböck, H. Schreiner and M. Stumptner, "Configuring large systems using generative constraint satisfaction," *IEEE Intelligent Systems,* vol. 13 (4), pp. 59-68, 1998.
[42] R. P. Desisto, "Constraints still key for product configurator deployments," Gartner, Inc., Stamford, CT, USA, Tech. Rep. T-22-9419, 1 June, 2004. 2004.
[43] A. Haag, U. Junker and B. O'Sullivan, "Explanation in product configuration," *IEEE Intelligent Systems,* vol. 22 (1), pp. 83-85, 2007.
[44] C. Sinz, A. Haag, N. Narodytska, T. Walsh, E. Gelle, M. Sabin, U. Junker, B. O'Sullivan, R. Rabiser, D. Dhungana, P. Grunbacher, K. Lehner, C. Federspiel and D. Naus, "Configuration," *IEEE Intelligent Systems,* vol. 22 (1), pp. 78-90, 2007.
[45] U. Junker and D. Mailharro, "Preference programming: Advanced problem solving for configuration," *AI EDAM,* vol. 17 (1), pp. 13-29, 2003.
[46] D. B. Searls and L. M. Norton, "Logic-based configuration with a semantic network," *The Journal of Logic Programming,* vol. 8 (1-2), pp. 53-73, 1990.
[47] F. Baader, "Description logics," in *Reasoning Web. Semantic Technologies for Information Systems*, S. Tessaris, E. Franconi, T. Eiter, C. Gutierrez, S. Handschuh, M. Rousset and R. A. Schmidt, Eds. Berlin, Heidelberg: Springer, 2009, pp. 1-39.
[48] J. R. Wright, E. S. Weixelbaum, G. T. Vesonder, K. E. Brown, S. R. Palmer, J. I. Berman and H. H. Moore, "A knowledge-based configurator that supports sales, engineering, and manufacturing at AT&T network systems," *AI Magazine,* vol. 14 (3), pp. 69-80, 1993.
[49] J. R. Wright, D. L. McGuinness, C. H. Foster and G. T. Vesonder, "Conceptual modeling using knowledge representation: Configurator applications," in *14th International Joint Conference on Artificial Intelligence (IJCAI-95), Workshop on Artificial Intelligence in Distributed Information Networks,* Montreal, Quebec, Canada, 1995, .
[50] D. L. McGuinness and J. R. Wright, "An industrial-strength description logic-based configurator platform," *IEEE Intelligent Systems,* vol. 13 (4), pp. 69-77, 1998.
[51] N. Sharma and R. Colomb, "Mechanising shared configuration and diagnosis theories through constraint logic programming," *The Journal of Logic Programming,* vol. 37 (1-3), pp. 255-283, 1998.
[52] T. Soininen, *An Approach to Knowledge Representation and Reasoning for Product Configuration Tasks.* Ph.D. thesis, Espoo, Finland: Helsinki University of Technology, Department of Computer Science and Engineering, 2000.
[53] T. Soininen, I. Niemelä, J. Tiihonen and R. Sulonen, "Representing configuration knowledge with weight constraint rules," in *AAAI Spring Symposium on Answer Set Programming: Towards Efficient and Scalable Knowledge (AAAI Technical Report SS-01-01),* Stanford University, CA, USA, 2001, pp. 195-201.
[54] G. Schenner, A. Falkner, A. Ryabokon and G. E. Friedrich. Solving object-oriented configuration scenarios with ASP. Presented at 15th International Configuration Workshop. 2013, Available: http://ws-config-2013.mines-albi.fr/CWS-2013-Proceedings-Color.pdf.
[55] A. Felfernig, L. Hotz, C. Bagley and J. Tiihonen, *Knowledge-Based Configuration: From Research to Business Cases.* Waltham, MA, USA: Morgan Kaufmann, 2014.
[56] A. Felfernig, L. Hotz, C. Bagley and J. Tiihonen, "Chapter 15 - configuration-related research challenges," in *Knowledge-Based Configuration*, A. Felfernig, L. Hotz, C. Bagley and J. Tiihonen, Eds. Boston: Morgan Kaufmann, 2014, pp. 191-195.
[57] C. Forza and F. Salvador, "Managing for variety in the order acquisition and fulfilment process: The contribution of product configuration systems," *Int J Prod Econ,* vol. 76 (1), pp. 87-98, 2002.
[58] C. Forza and F. Salvador, "Product configuration and inter-firm coordination: an innovative solution from a small manufacturing enterprise," *Comput. Ind.,* vol. 49 (1), pp. 37-46, 2002.
[59] T. Blecker, G. Friedrich, B. Kaluza, N. Abdelkafi and G. Kreutler, *Information and Management Systems for Product Customization.* Boston: Springer, 2005.
[60] M. Heiskala, K. Paloheimo and J. Tiihonen, "Mass customization with configurable products and configurators: A review of benefits and challenges," in *Mass Customization Information Systems in Business*,

1st ed., T. Blecker and G. Friedrich, Eds. Hershey, PA, USA & London, UK: IGI Global, 2007, pp. 1-32.
[61] F. Salvador and C. Forza, "Principles for efficient and effective sales configuration design," *International Journal of Mass Customisation,* vol. 2 (1-2), pp. 114-127, 2007.
[62] C. Forza and F. Salvador, *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, 2006.
[63] A. R. Hevner, S. T. March, J. Park and S. Ram, "Design science in information systems research," *MIS Quarterly,* vol. 28 (1), pp. 75-105, 2004.
[64] S. Gregor, "The nature of theory in information systems," *MIS Quarterly,* vol. 30 (3), pp. 611-642, 2006.
[65] A. Anderson and M. Pasanen, "WeCoTin Requirements and architecture (unpublished)," 2003.
[66] J. Tiihonen, T. Soininen, I. Niemelä and R. Sulonen, "A practical tool for mass-customising configurable products," in *Proceedings of the 14th International Conference on Engineering Design,* Stockholm, Sweden, 2003, pp. CDROM, paper number 1290, 10 pp.
[67] F. Salvador, P. M. de Holan and F. T. Piller, "Cracking the code of mass customization," *MIT Sloan Management Review,* vol. 50 (3), pp. 71-78, 2009.
[68] P. Simons, I. Niemelä and T. Soininen, "Extending and implementing the stable model semantics," *Artif. Intell.,* vol. 138 (1-2), pp. 181-234, 2002.
[69] A. Felfernig, M. Mandl, J. Tiihonen and M. Schubert. Personalized product configuration. Presented at Multikonferenz Wirtschaftsinformatik 2010, 24. PuK-Workshop: Planung/Scheduling Und Konfigurieren/Entwerfen. 2010, Available: http://webdoc.sub.gwdg.de/univerlag/2010/mkwi/.
[70] J. Tiihonen, "Characterization of configuration knowledge bases," in *19th European Conference on Artificial Intelligence (ECAI-2010), Workshop on Intelligent Engineering Techniques for Knowledge Bases (IKBET),* Lissabon, Portugal, 2010, pp. 13-20.
[71] J. Tiihonen, T. Soininen, I. Niemelä and R. Sulonen, "Empirical testing of a weight constraint rule based configurator," in *15th European Conference on Artificial Intelligence (ECAI-2002), Configuration Workshop,* Lyon, France, 2002, pp. 17-22.
[72] J. Tiihonen, *Computer-Assisted Elevator Configuration.* M.Sc (Eng.) thesis, Espoo: Helsinki University of Technology, Department of Computer Science, 1994.
[73] T. Soininen and J. Tiihonen, "Sales configurator in Datex procuct data management process," 1995.
[74] J. Tiihonen and T. Soininen, "Product configurators - information system support for configurable products," Helsinki University of Technology, Espoo, Tech. Rep. TKO-B 137, 1997.
[75] J. Tiihonen, *National Product Configuration Survey — customer Specific Adaptation in the Finnish Industry.* Licentiate of technology (Eng.) thesis, Espoo: Helsinki University of Technology, Department of Computer Science, Laboratory of Information Processing Science, 1999.