Proceedings of the ECAI 2010 Workshop on Intelligent Engineering Techniques for Knowledge Bases (IKBET)
affiliated with the 19th European Conference on Artificial Intelligence
August 16-20, 2010. Lisbon, Portugal.
Alexander Felfernig, Franz Wotawa (eds.)

Characterization of configuration knowledge bases

Juha Tiihonen
Department of Computer Science and Engineering
Aalto University
Juha.Tiihonen@tkk.fi

Abstract

Characterization of configuration problems has remained limited. This work characterizes 26 configuration models with numerous indicators of size and degree of application of modeling mechanisms including inheritance and application of advanced compositional structure. The original goal of modeling was to evaluate and demonstrate the applicability of WeCoTin configurator and PCML modeling language to industrial configuration problems. The main contribution of the paper is in providing probably the first multi-faceted detailed characterization of a relatively large number of configuration problems. Additionally, aspects for characterizing configuration models were identified.

1 Introduction

Due to active research in the configuration domain, several formalisms for representing configuration knowledge have been proposed, and configurators are used to support day-to-day business in many companies. However, characterization of practical configuration problems has remained limited in the literature. Of course, a number of individual cases have been documented thoroughly, most importantly the R1/XCON [Barker, et al. 1989] and the VT/Sisyphus elevator configuration problem [Schreiber, et al. 1996], not forgetting characterizing the domain and configuration models when describing configurator implementations, e.g. [Fleischanderl, et al. 1998]. Further, classification of configuration tasks has been proposed [Wielinga, et al. 1997, Haag. 2008], which requires characterization of the tasks as a basis for classifications.

There are several reasons why configuration tasks should be characterized. These include enabling evaluation of effectiveness of specific representation formalisms and modeling constructs, gaining understanding on the nature of different configuration tasks, which in turn could enable supporting tools that better match practical problems and facilitates development of benchmarks, and enabling classification of configuration tasks. Configuration models, related configuration tasks and their IT support could be characterized from several perspectives. These include the size of the models, computational performance, and complexity, effectiveness of specific modeling techniques related to specific configuration tasks as defined by the company offering. Less technical views cover aspects of practical interest such as the proportion of cases covered by the configuration models, completeness of the models in terms of business requirements, usability and different aspects of utility provided and sacrifices required.

This work reports a part of evaluation of the WeCoTin configurator [Tiihonen, et al. 2003] and especially its modeling capabilities. 26 configuration models were created to evaluate and demonstrate the applicability of WeCoTin and PCML to real industrial configuration problems, which demonstrates the high-level efficacy of the constructs. The models will be characterized with numerous indicators of size and degree of application of different modeling constructs. Five tables provide characterizations models.
This paper is structured as follows. In Section 2 the applied modeling language will be described. Section 3 gives an overview of the configuration models and their background. Section 4 describes component type hierarchy, overall configuration model size, and price modeling. Section 5 details the compositional structure of the models, Section 6 attributes, and Section 7 constraints. Finally, discussion, future work and conclusions are presented in Section 8.

2 Product Configuration Modeling Language

The configuration models have been modeled with ***PCML*** (Product Configuration Modeling Language) [Tiihonen, et al. 2003, Peltonen, et al. 2001]. PCML is object-oriented, declarative and it has formal implementation-independent semantics. The semantics of PCML is 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 short hand notations for a set of sentences in the weight constraint rule language (***WCRL***) (Soininen et al. 1998). PCML covers a practically important subset of a synthesized ontology of configuration knowledge [Soininen, et al. 1998].

The main concepts of PCML are ***component types***, their ***compositional structure***, ***properties*** of components, and ***constraints***. Component types define the parts and

Model	Company & Description Status and validation
1 Compressor FM	Gardner Denver. Compressor family FM series. 18-40kW compressors. Internal view for sales persons. Demonstrated integration to automatic manufacturing completion (EDMS2), and e-commerce site Intershop 4. Delivery time calculation. Complete model. Demonstrations to company with expert and focus group validation on matching company view of product configurability. Company workers had a few dozen configuration sessions over the web. Model used in empirical performance testing. Manually analyzed the number of possible configurations, which matches the number of answer sets (configurations) generated by the inference engine.
2Compr FM sc	Gardner Denver. Model 1 Compressor FM above augmented with 13 pre-selection packages to represent agreed-on customer standards with their defaults, usually enforced with soft constraints. Behaves as intended.
3 Compr FS	Gardner Denver compressor family, 11-18 kW FS series compressors. Internal view for sales persons. Complete model. Demonstrations to company with expert and focus group validation on matching company view of product configurability. In addition, company workers had configuration sessions over the web. Model used in empirical performance testing. The manually analyzed number of possible configurations matches the number generated by the inference engine.
4 Compr FX	Gardner Denver compressor family, 4-10kW FX series compressors. Internal view for sales persons. Complete model. Demonstrations to company with expert and focus group validation on matching company view of product configurability. In addition, company workers had configuration sessions over the web. Model used in empirical performance testing. The manually analyzed number of possible configurations matches the number generated by the inference engine.
5Compr FL	Gardner Denver compressor family, 45-80 kW FL series compressors. Internal view for sales persons. Complete model, seems to work, no validation in company.
6Compr M	Gardner Denver compressor family. 75-160kW M series compressors. Internal view for sales persons. Complete model, seems to work, no validation in company.
7 KONE old	KONE maintenance service contracts, older company offering. Some additional options that can be specified after contract made (billing options, e-notification). Some one-time service extras like long-term maintenance plan, condition check. Extensive on-line help texts for salespersons developed together with company, shown with the description mechanism. Complete model with extras. Demonstrations to company with expert and focus group validation on matching company view of product configurability. Installed to product manager computer for test use.
8 KONE new	KONE maintenance service contracts. Options of newer maintenance service contracts. Additional options that can be specified after contract made (billing options, extensive e-services). Some one-time service extras like long-term maintenance plan, condition check. Extensive on-line help texts for sales persons. Complete model with extras. Demonstrations to company with expert and focus group validation on matching company view of product configurability. Installed to product manager computer for test use.
9 Bed	Not public 1. Real hospital bed product line based on order forms and additional information. Complete model. Demonstration to company and focus group on matching company view of product configurability.
10Fireplace	Not public 2. Modular fireplace. Technology demonstration with CAD vendor. Simple product. Integrated with 3D CAD to visualize configuration changes. Complete model. Validation with CAD vendor.
11 Patria Pasi	Patria Vehicles. Military vehicle. For company internal systematic documentation of available productized options. Complete model with respect to standardized offering. Demonstrations to company with expert and focus group validation on matching company view of product configurability. Configuration model validated in internal test use in company.
12 Dental	Not public 3. Real integrated dental unit and patient chair. Most difficult to configure product of the company. Complete model. Demonstration to company, brief focus group, expert validation.
13 X-ray	Not public 3. Real X-ray unit for dentists. Product designed with ease of configuration in mind. Complete model. Demonstration to company, brief focus group, expert validation.
14 Vehicle	Not public 4. Self-moving machine industry product. Real product based on order forms and interviews. Partial model for demonstration purposes representing about half of the sales view of the product. Numerous optional parts and some simple constraints were excluded. Despite omissions the model reflects quite well the nature of sales configuration of this vehicle. Test-used by company stakeholders over the web. Functionality found satisfying "better than our commercial product". The manually analyzed number of possible configurations matches the number generated by the inference engine.
15 Insur 1	Tapiola group. Insurance coverage for families (persons, travel, car, home, cottage) as a combination of insurance products. Demonstration model with a subset of the whole offering, Discretized large domain specification attributes. Demonstrations to company.
16 Insur 2	Tapiola group. Comprehensible insurance coverage for family's person related risks. Non-traditional risk-oriented (not insurance product oriented) way of asking which coverage is desired. These selections are satisfiable with a combination of real products. Mapping to products not performed in model. Demonstration model. Demonstration to company and focus group.
17 Insur 3	Tapiola group. Experimented 4-world model objects-world questions. Solution-world with detailed model of real offered car-related insurance coverage. HUT internal, no company validation.
18 Insur 4	Tapiola group. Comprehensive insurance coverage for families and property. Tested some 4-worlds model ideas. Objects: persons, home, leisure-time apartment, vehicles (cars, motorcycle, boat), forest, domestic animals (dogs, horses, cats). Solutions corresponding real insurance products and their availability, modeled in detail. However, risks coverage for home and cottage is selectable with excessively small granularity. HUT internal, no company validation.

19 Mob Subscr 1	Elisa (Radio-linja). Demonstration on configuring mobile subscription and its value-added services while taking into account phone capabilities. Model covers only aspects considered interesting for the demonstration. Information acquired from public company web-site. Demonstration model, demonstrated to company stakeholders.
20 Mob Subscr 2	Elisa. Real mobile subscription + phone bundle as basis. Reverse-engineered from company web site + added needs analysis questions to identify suitable product options with soft constraints HUT internal validation against offering., presentation in an open seminar for the Finnish industry.
21 Mob Subscr 3	Telia-Sonera. Demonstration on selecting mobile subscription and some value-added services based on usage characteristics. Implemented with hard and soft constraints. Based on public information from company website. No validation. Behaves as intended by the modeler.
22 Broadband	Telia-Sonera. Real broadband subscription product line. 4 worlds model demonstration. Re-engineered from company website, added customer and needs analysis questions, and aspects of delivery process that are configured based on selected options. Soft constraints warn when selections do not correspond to needs. Complete model with extras. Demonstration to the company.
23 Linux	Debian Linux Familiar distribution configuration model over 6 points of time and several software versions. Information gathered from Debian Linux version compatibility lists, configuration model generated via script by mapping package descriptions into PCML [Kojo, et al. 2003]. A newer version than that in Kojo et al. was characterized. No validation, "seems and behaves right", although with slow performance.
24 Iced	None. Demonstration: minimal fictive car model for ICED conference article describing WeCoTin configurator. No validation. Behaves as intended by the modeler.
25Wecotin car	BMW. Demonstration model based on a subset of a real car, identified configuration rules from company website No validation. Behaves as intended by the modeler.
26 CarDiss	BMW. Demonstration model based on 25 WeCoTin Car, with some extra fictive features to demonstrate higher cardinality. No validation. Behaves as intended by the modeler.

Table 1. Identification, description, status and validation of the configuration models.

properties of their ***individuals*** that can appear in a configuration. A component type defines its compositional structure through a set of ***part definitions***. A part definition specifies a ***part name***, a non-empty set of ***possible part types*** (***allowed types*** for brevity) and a ***cardinality*** indicating the possible number of parts. A component type may define properties that parametrize or otherwise characterize the type. A ***property definition*** consists of a ***property name***, a ***property value type*** and ***a necessity definition*** indicating if the property must be given a value in a complete configuration. Component types are organized in a ***class hierarchy*** where a ***subtype*** inherits the property and part definitions of its ***supertypes*** in the usual manner. When a type inherits data 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 component type is either ***abstract*** or ***concrete***. Only an individual directly of a concrete type can be used in a configuration. Constraints associated with component types define conditions that a correct configuration must satisfy. A constraint expression is constructed from references to parts and properties of components and constants such as integers. These can be combined into complex expressions using relational operators and Boolean connectives.

3 Model background, identification, characterization, status and validation

PCML and WeCoTin have been used to model and configure the complete sales view of 14 real products or services, and partial sales view of 8 products or services. In the complete sales views all known configurable options of the products or services have been modeled. Configuration models of some products or services contained extra features that are not normally taken into account during sales configuration. Three additional demonstration models are included in the characterizations. Some configuration models were created in early phases of WeCoTin construction, some after completion of the development project.

In most cases, order forms, brochures, and other documentation were used as a basis for modeling, and company representatives were contacted for additional information before showing the results as demonstrations. In some cases it was possible to re-engineer configuration model information from company websites.

The WeCoTin modeling tool was instrumented to provide the characterizing metrics based on static configuration model analysis presented in Tables 2-5. The configuration models come from the following domains:

- Eight models are from machine industry and come from three companies (5 compressors (1 twice), an undisclosed vehicle, and one military vehicle).
- Three models from two companies are from healthcare domain (2 dentist equipment product families, a hospital bed family.)
- Four models from two companies are from telecommunications domain (3 mobile and 1 broadband subscriptions).
- Three models from one company are from insurance domain.
- Two models from one company represent two generations of maintenance contracts of elevators.

- One model is software configuration (Debian Linux Familiar with package versions).
- One model demonstrates configuration of a modular fireplace.
- Three models are pure demonstration models – two are based on a subset of a real car, and one is fictional.

Table 1 identifies the models and briefly characterizes the domain of each configuration model. Each configuration model is identified with a unique numeric identifier that remains the same in each table. Model names have been abbreviated in later tables due to space constraints.

Table 2 details the degree of configuration model completeness and model validation status. Five models from three companies have been test-used by company representatives. In addition, configuration demonstrations and immediately following focus groups have been used for validation of additional seven configuration models.

4 Taxonomy, model size, and pricing

Table 2 provides characterization of component type hierarchy used in the models, overview of model size, and .information on price modeling.

One model (23, Linux) was significantly larger than others and semi-automatically generated. Discussions on model characterizations exclude this model, but averages and totals are calculated with and without it.

Numbers of abstract, concrete, and total component types contribute to the size of a configuration model, and are shown in corresponding columns of Table 2. The total number of component types varied from one 1 to 626, the median was 9 and average was 18 (42 with Linux). The number of direct subtypes of abstract types (other than root of component type hierarchy `Component`) ("Subtypes") characterizes the number of component types organized in a type hierarchy. Interpreted as a percentage "% as subtypes", the figure varied from 0% to 100%, with average without Linux being 59% and median 46%.

Each selectable attribute or part of a component individual being configured generates a ***question*** during a configuration process. The number of questions in a configuration model ("Questions") roughly characterizes the size of each configuration model and the related configuration task. In a typical configuration model without redundant concrete component types, each question might have to be answered while configuring a product. All possible questions may not be asked in a configuration session, because an individual of a specific type is not necessarily selected into a configuration, or if some attributes or parts are defined to be invisible to the user or to have a fixed value. On the other hand, if several individuals of a component type are in a configuration, the number of questions may be multiplied. The average was 61 questions per configuration model, and roughly 5.4 questions per concrete type (excluding Linux).

Model	Total types	Abstract types	Concrete types	Subtypes	% as subtypes	Questions	% questions in root	Constraints	Price
1 C FM	9	2	7	4	44	31	58	17	adv
2 CFm sc	9	2	7	4	44	31	58	17	adv
3 C FS	3	0	3	0	0	24	88	14	adv
4 C FX	1	0	1	0	0	20	100	23	adv
5 C FL	9	2	7	4	44	28	64	13	no
6 C M	3	0	3	0	0	23	91	14	no
7 KO old	5	0	5	0	0	28	79	13	no
8 Ko new	15	3	12	7	47	77	4	1	no
9 Bed	31	8	23	27	87	34	76	10	basic
10 Firepl	7	1	6	4	57	4	75	0	no
11 Pasi	5	1	4	2	40	79	95	13	no
12 Dental	64	11	53	43	67	109	3	36	no
13 X-ray	11	2	9	4	36	37	41	3	no
14 Vehicl	28	4	24	9	32	24	75	7	basic
15 Ins 1	8	2	6	5	63	30	20	4	no
16 Ins 2	62	13	49	56	90	49	20	0	no
17 Ins 3	11	3	8	5	45	41	29	14	no
18 Ins 4	37	11	26	34	92	242	5	84	no
19 Mob 1	4	0	4	0	0	18	56	6	basic
20 Mob 2	39	9	30	38	97	65	25	28	basic
21 Mob 3	5	1	4	3	60	21	38	6	no
22 Broad	66	15	51	64	97	485	1	43	no
23 Linux	626	1	625	624	100	4369	14	2380	no
24 Iced	8	2	6	5	63	4	75	3	basic
25 Wcar	6	1	5	2	33	10	60	3	basic
26CarDis	10	2	8	5	50	12	58	3	basic
Total	1082	96	986	949		5985		2755	
Total no Linux	456	95	361	325		1526		375	
Average	42	4	38	37	48	227	50	106	
Avg no Linux	18	4	14	13	59	61	52	15	
Median	9	2	7	5	46	31	58	13	
Min	1	0	1	0	0	4	1	0	
Max	626	15	625	624	100	4369	100	2380	

Table 2. Use of pricing mechanisms, company look, and component type hierarchy in the configuration models.

Especially simpler models were often centered on the configuration type that is the root of the compositional hierarchy. The degree of such concentration is characterized by the proportion of questions defined in the configuration type. Column "% questions in root" specifies this proportion. On the average about half (50%), and median 58% of questions were in the root component type, with a large scale of variation.

The total number of constraints ("Constraints") specified in the component types of each configuration model varied

largely, but usually remained in a few dozen at maximum. More details will be given in Section 2.5.

Column “Price” indicates which of two price calculation mechanisms, if any, was used. The “basic” mechanism considers additive prices: each component individual can have a base price determined by its type, and each attribute value can specify additional price. The sum of component individual and their attribute value prices is the price of the configuration. Four real products and three demonstration models applied this pricing mechanism.

A more advanced calculation mechanism (“adv”) [Nurmilaakso. 2004] performs definable calculations as function of the current configuration, configuration model, and external data. Values are provided to calculations when condition expressions examining a configuration evaluate to true. Three products were priced with this mechanism.

Prices were often omitted either due to indicated sensitivity or to constrain resource usage. The simple mechanism would have been sufficient for other products except compressors and insurance products.

5 Compositional structure

Table 3 exhibits details of applying compositional structure in the modeled cases. An indication on the number of parts in a configuration model is given by the number effective parts in concrete types of the model. The number of effective parts (“Effective parts”) is the sum of inherited and locally defined parts in concrete types. The average number of effective parts per concrete component type (“Eff. parts / concrete”) characterizes the breath of the configuration tree and average application of the compositional structure as a modeling mechanism. On the average, 10 parts were defined in each configuration model. Median was 4. The average of 0.6 effective parts in each concrete type indicates moderate usage of the compositional structure.

Inheritance in the compositional structure was used on the average less frequently than direct part definitions in concrete types: Eight models applied inheritance of parts whereas parts were introduced in 25 models. On the average, seven part definitions were defined in concrete types (“Part def in concrete”), and one part per configuration model was defined in abstract types (“Part def in abstract”). Four of the effective 10 part definitions were inherited. Three inherited part definitions were applied as such (“Non-refined inherited”), and one was refined, e.g. to restrict the set of allowed types (“Refined inherited”). Some models applied part inheritance significantly more. For example, in model 12 Dental, 26 (79%) of 33 effective part definitions were inherited (“% inherited parts”). Out of these 6 were refined. The eight models applying part inheritance had 31% (80) of their 257 effective part definitions inherited.

Many configuration models concentrated part definitions on the configuration type. An average configuration type contained 4 part definitions (“Part def in conftype”). In 12 models all parts were defined in the configuration type. The average percentage of part definitions in the configuration type was 70% (“% part def in conf type”).

The cardinality of a part definition defines how many component individuals must realize the part in a consistent and complete configuration. On the average, 6 part definitions were optional, that is, with cardinality with 0 to 1 (“0 to 1 cardinality”), and 2 part definitions were obligatory with cardinality 1 to 1 (“1 to 1 cardinality”). Only one demonstration model contained one part definition with a larger maximum cardinality than one (“max cardinality 2+”).

Model	Effective parts	Eff. parts / concrete	Part def in concrete	Part def in abstract	Part def in conf type	% part def in conftype	Non-refined inherited	Refined inherited	% inherited parts	0 to 1 cardinality	1 to 1 cardinality'	max cardinality 2+	Enumerated allowed	Effective allowed	% allowed saved	Max allowed
1 C FM	4	0.6	2	1	2	50	1	1	50	0	3	0	6	6	0	2
2 C Fm sc	4	0.6	2	1	2	50	1	1	50	0	3	0	6	6	0	2
3 Com FS	1	0.3	1	0	1	100	0	0	0	0	1	0	2	2	0	2
4 com FX	0	0.0	0	0	0	-	0	0	-	0	0	0	0	0	-	0
5 com FL	4	0.6	2	1	2	50	1	1	50	0	3	0	6	6	0	2
6 com M	1	0.3	1	0	1	100	0	0	0	0	1	0	2	2	0	2
7 KO old	2	0.4	2	0	2	100	0	0	0	1	1	0	4	4	0	2
8 Ko new	19	1.6	10	3	2	11	5	4	47	11	2	0	14	17	18	3
9 Bed	3	0.1	3	0	3	100	0	0	0	0	3	0	5	32	84	12
10 Firepla	2	0.3	2	0	2	100	0	0	0	1	1	0	2	5	60	4
11 Pasi	2	0.5	2	0	2	100	0	0	0	1	1	0	3	3	0	2
12 Dental	33	0.6	7	13	1	3	20	6	79	18	2	0	33	80	59	8
13 X-ray	5	0.6	3	1	2	40	0	2	40	3	1	0	8	8	0	3
14 Vehicl	16	0.7	16	0	12	75	0	0	0	12	4	0	20	24	17	3
15 Insur 1	10	1.7	10	0	6	60	0	0	0	9	1	0	10	10	0	1
16 Insur 2	30	0.6	22	4	10	33	8	0	27	24	2	0	26	29	10	4
17 Insur 3	12	1.5	12	0	11	92	0	0	0	10	2	0	13	13	0	2
18 Insur 4	53	2.0	30	5	12	23	27	0	51	35	0	0	35	46	24	4
19 Mob 1	3	0.8	3	0	3	100	0	0	0	3	0	0	3	3	0	1
20 Mob 2	13	0.4	13	0	12	92	0	0	0	8	5	0	15	27	44	5
21 Mob 3	1	0.3	1	0	1	100	0	0	0	1	0	0	3	3	0	3
22 Broad	32	0.6	30	1	4	13	2	0	6	10	21	0	32	53	40	15
23 Linux	624	1.0	624	0	624	100	0	0	0	624	0	0	624	624	0	1
24 Iced	2	0.3	2	0	2	100	0	0	0	0	2	0	2	5	60	3
25 Car	2	0.4	2	0	2	100	0	0	0	1	1	0	3	4	25	2
26 CarDis	3	0.4	3	0	3	100	0	0	0	1	1	1	4	7	43	3
Total	881		805	30	724		65	15		773	61	1	881	1019		
Total no Linux	257		181	30	100		65	15		149	61	1	257	395		
Average	34	0.7	31	1	28	72	3	1	16	30	2	0	34	39	19	4
Avg no li	10	0.6	7	1	4	70	3	1	17	6	2	0	10	16	20	4
Median	4	0.6	3	0	2	92	0	0	0	1	1	0	6	7	0	3
Minimum	0	0.0	0	0	0	3	0	0	0	0	0	0	0	0	0	0
Maximum	624	2.0	624	13	624	100	27	6	79	624	21	1	624	624	84	15

Table 3. Compositional structure of the configuration models.

A part is realized with individuals(s) of allowed types. A part definition explicitly enumerates the allowed types. The number of directly enumerated allowed types ("Enumerated allowed") is often smaller than the number of effectively allowed component types ("Effective allowed"), because specifying a supertype as an allowed type effectively specifies the concrete subtypes as allowed types. An average configuration model directly specified 10 and effectively 16 component types in the average 8 part definitions. The average percentage of "savings" was 20% ("% allowed saved"). The relatively low number of allowed types is partially explained by relatively often occurring optional parts with only one effective allowed type. The maximum number of effective allowed types in a part definition ("Max allowed") was on the average and median 4 types, and maximally 15.

6 Attributes

Table 4 exhibits details of applying attributes in the modeled cases. A rough indication on the number of attributes in a configuration model is given by the number effective attributes of the configuration model. The number of effective attributes ("Effective attributes") is the sum of inherited and locally defined attributes in concrete types. Concrete types had a total of 1269 effective attributes. Average without the Linux model was 51 effective attributes and the median was 25. The average of average number of effective attributes per concrete component type ("Effective / concrete") was 4.8, and median of averages was 3.7.

Inheritance of attributes was applied more frequently than inheritance of parts, but less frequently than direct attribute definitions in concrete types: 16 models applied inheritance of attributes whereas all 26 models defined attributes.

On the average a model contained 26 attribute definitions in concrete types ("Defs in concrete"). The 5 attribute definitions in abstract types ("Defs in abstract") expanded to an average of 25 effective attributes in concrete types. The average percentage of inherited attributes in concrete types ("% Inherited") was 23%. Some models applied attribute inheritance more significantly, e.g. 44 - 89% of effective attributes were inherited in some larger models.

Of the 1269 effective attributes, 51% (642) were defined locally ("Defs in Concrete"), and the remaining 49% (627) were inherited. 122 attribute definitions in abstract types ("Defs in abstract") were inherited as such into 537 attributes in subtypes ("Non-refined inherited"), and into 168 attributes in refined form ("Refined inherited"), a total of 705 inherited attributes.[1] Often attribute definitions were concentrated on the configuration type, 54% (14) of the 26 models specified at least 50% of effective attribute definitions there. An average configuration type defined 11 attributes ("Def in config type"). The average percentage of attribute definitions in the configuration type was 46%. ("% part def in conf type").

[1] The 705 inherited attributes includes 78 (705-627) attributes inherited to abstract types.

Model	Effective attributes	Effective / concrete	% Inherited	Attr. definitions	Defs in concrete	Defs in abstract	Def in config type	Boolean	Enumerated string	Integer	Refined inherited	Non-refined inherited	Optional attr. defs	Maximum domain	Domain 1	Domain 2 to 3	Domain4 to 10	Domain 11+
1 FM	27	3.9	22	24	21	3	16	7	13	4	2	4	0	61	0	17	6	1
2 Fm sc	27	3.9	22	24	21	3	16	7	13	4	2	4	0	61	0	17	6	1
3 FS	23	7.7	0	23	23	0	20	5	13	5	0	0	0	51	0	16	6	1
4 FX	20	20.0	0	20	20	0	20	5	10	5	0	0	0	44	0	12	7	1
5 FL	24	3.4	17	22	20	2	16	7	12	3	2	2	0	20	0	13	7	1
6 M	22	7.3	0	22	22	0	20	5	14	3	0	0	0	15	0	11	8	1
7 K old	26	5.2	0	26	26	0	20	8	9	4	0	0	0	10	0	14	7	0
8 k new	58	4.8	81	29	11	18	1	12	7	2	23	24	0	4	0	18	3	0
9 Bed	31	1.3	0	31	31	0	23	17	13	1	0	0	2	7	0	26	5	0
10 Fire	2	0.3	0	2	2	0	1	0	2	0	0	0	0	2	0	2	0	0
11 Pasi	77	19.3	3	76	75	1	73	39	37	0	0	2	0	5	0	67	9	0
12 dent	76	1.4	70	48	23	25	2	28	19	1	3	50	8	10	0	41	7	0
13 xray	32	3.6	44	25	18	7	13	10	7	8	2	12	2	11	0	16	1	8
14Vehi	8	0.3	0	8	8	0	6	3	4	1	0	0	0	22	0	5	2	1
15 Ins1	20	3.3	10	19	18	1	0	4	6	9	0	2	0	11	1	9	8	1
16 Ins2	19	0.4	58	12	8	4	0	4	2	5	0	11	0	11	0	7	3	1
17 Ins3	29	3.6	0	29	29	0	1	17	6	4	0	0	3	122	0	21	4	2
18 Ins4	189	7.3	26	159	140	19	0	144	3	2	0	49	1	101	0	146	1	2
19 Mo1	15	3.8	0	15	15	0	7	10	2	3	0	0	2	13	0	10	2	3
20 Mo2	52	1.7	29	40	37	3	4	25	10	2	15	0	1	5	0	35	2	0
21 Mo3	20	5.0	60	12	8	4	7	12	0	0	0	12	0	2	0	12	0	0
22 Broa	453	8.9	89	81	51	30	0	51	12	3	119	361	1	436	4	58	1	3
23 Linu	3745	6.0	67	1253	1249	4	1	0	1248	3	1248	1248	0	6	551	691	9	0
24 Iced	2	0.3	0	2	2	0	1	0	1	1	0	0	0	2	0	2	0	0
25Wcar	8	1.6	25	7	6	1	4	4	3	0	0	2	0	5	0	6	2	0
26 Diss	9	1.1	22	8	7	1	4	5	3	0	2	2	0	5	0	6	2	0
Total	5014			2017	1891	126	276	429	1469	73	1416	1785	20		556	1278	108	27
Tot. no Linux	1269			764	642	122	275	429	221	70	168	537	20		5	587	99	27
Average	193	4.8	25	78	73	5	11	17	57	3	54	69	1	40	21	49	4	1
Avg no Linux	51	4.8	23	31	26	5	11	17	9	3	7	21	1	41	0	23	4	1
Median	25	3.7	19	24	21	1	5	7	8	3	0	2	0	11	0	15	4	1
Minimum	2	0.3	0	2	2	0	0	0	0	0	0	0	0	2	0	2	0	0
Maximum	3745	20.0	89	1253	1249	30	73	144	1248	9	1248	1248	8	436	551	691	9	8

Table 4. Attributes in the configuration models.

Attribute value types were distributed as follows: of average 31 attribute definitions per configuration model, 17 were Boolean ("Boolean"), 9 were enumerated strings (""Enumerated string"), 3 integers ("Integer"), and 2 unconstrained strings. In total there were 56% (429)

Boolean, 29% (221) enumerated string, 9% (70) integer, and 6% (44) unconstrained string attribute definitions. Unconstrained strings specified additional details such as customer names, addresses, etc. aspects that do not require inference.

Attribute domain sizes remained quite small. A domain of at least 11 possible values ("Domain 11+") was present in about 4% (27) of 717 attribute definitions with a fixed domain. The maximum domain size ("Maximum domain") varied significantly – the largest domain was 436 possible values, in this case enumerated string values. The most common domain size was 2 to 3 possible values ("Domain 2 to 3") in 82% (587) of attribute definitions. 14% (108) of attribute domains were of size 4-10 ("Domain 4 to 10"). Domain of size 1 ("Domain 1") was encountered in 1% (5) cases.

20 attributes were defined as ***optional***, ("Optional attr. defs") meaning that it is possible to specify in a complete configuration that no value will be assigned to the attribute.

7 Constraints

The number of constraints varied significantly from 0 to 84 (2380 with Linux), an average of 15 per model. The median was 13. On the average, two of the constraints were soft, and the rest were hard. Totally 44 constraints were defined in abstract component types, of these 40 were hard and 4 soft. As with other modeling constructs, definition of a constraint in a supertype causes inheritance to subtypes.

It is not trivial to characterize complexity of constraints. A simple syntactic metric based on parse tree complexity of the resulting constraint expression was calculated. For example, consider the following a constraint:

```
Active_Cruise_Control_Requires_BiXenon
( Cruise_control = true ) implies
($config.Headlights individual of BiXenon)
```

The "complexity" of the example constraint is seven (7). Complexity of a literal, a constant, a variable, a component type reference, element access, an ID-expression, or an element reference in the expression is one. Each operator application counts one plus complexity of each argument.

Typical constraints were small, almost half (45%) of the constraints were of roughly the same complexity as the example constraint above, and 36% a bit more complex.

- 12% (45) of constraints had complexity 0-5
- 45% (170) of constraints had complexity 6-10
- 36% (135) of constraints had complexity 11-20
- 4% (14) of constraints had complexity 21-50
- 1% (4) of constraints had complexity 51-100
- 1% (4) of constraints had complexity 101-1000
- 1% (3) of constraints had complexity over 1000

Maximum constraint complexity varied significantly. The median was as low as 13, and average without Linux was 235. The maximum complexity was 1319. All the compressor models had a large table constraint specifying feasible combinations of values of 5 attributes, each with a relatively large number of rows, which explains the high average.

8 Discussion, future work, and conclusions

8.1 Limitations

Modeling and evaluation of modeling mechanisms contains author bias. All modeling was performed by researchers who were involved in development of the system.

The companies whose products were modelled were either existing or potential research partners. In other words, the sample of companies was not selected e.g. to cover most challenging cases such as telecommunications networks.

8.2 Modeling mechanisms

Some partial models were created due to resource constraints, or when the purpose of modeling was attainable with partial modeling. In other words, capabilities of PCML or WeCoTin did not limit the scope of modeling. However, Floats, fixed point numbers or integers with very large domain would have been useful in the insurance and compressor domains. In the insurance case some specification variables such as a desired amount of monetary coverage were specifiable with arbitrary monetary amounts, which can lead to very large domains. Apartment size and desired coverage were discretized in model "16 Insur 1". In compressor models, the company had calculated and validated combinations of specification variable values that produce a specific nominal capacity, represented in a table constraint. Further, a specific percentage of capacity loss is encountered in high altitude use environments. Calculating this would have been more convenient with floating or fixed point arithmetic.

Application of the compositional structure was important but less frequent than anticipated. A partial explanation is that it was often considered more practical to model alternative or optional components as enumeration or Boolean attributes rather than as a part, if there was no need to configure details of the selected component individuals.

Part definitions with cardinality were useful: the mechanism provides a convenient way to model selecting at most one or exactly one component individual to a role in product structure out of several alternatives. This capability prevents the need for a number of extra constraints. For example, some commercial systems require that each alternative is specified as optional, and a mutual exclusivity constraint is required [Damiani, et al. 2001]. However, sometimes the mechanism was a bit clumsy: in case of an optional part (cardinality 0 to 1), and exactly one allowed type, it was difficult to invent a name for the component type and for the part. A bit surprisingly, large cardinalities were not needed in these configuration models.

Applying inheritance saved modeling effort in larger models significantly. Almost half (49%) of effective attributes were inherited, and one definition in a supertype created in average 4.4 effective attributes. Refinement of inherited attributes and parts was a useful mechanism for

limiting the domain of allowed values or allowed types. created through statistics. Refinement facilitated the application of inheritance also in cases where some subtypes have a narrower range of allowed values or types. Inheritance related to compositional structure was also useful, but was applied only in about 31% of the models. This mechanism was generally used in larger models, where almost a third of part definitions were inherited.

There was no need for explicit resource balancing or satisfaction in the modeled cases. There was no need for topological modeling, e.g. ports in our modeled cases. However, when modeling services and their delivery processes [Tiihonen, et al. 2007], there was a need to assign different stakeholders as resources that participate in different service activities. This assignment can be somewhat clumsily modeled with attributes. However, allocation of responsibilities to different, dynamically defined stakeholders could be more naturally modeled as connections between the activities and stakeholders.

8.3 Future work

The amount of work required to create a configuration model depended to a large extent on the knowledge acquisition and validation work. Collecting reliable statistics on total effort of creating and maintaining configuration models remains future work.

Performance evaluations are important characterization of configuration models. In previous work, performance of some of the models has been evaluated, and was found satisfactory[Tiihonen, et al. 2002]. However, performance should be tested with a larger and more representative set of configuration models.

8.4 Conclusions

The main contribution of this paper is providing, to our knowledge, the first multi-case in-depth characterization of configuration models and analysis of utility of modeling mechanisms. A combination of taxonomic hierarchy with inheritance and strict refinement, compositional structure with the concept of part definitions, attributes, and constraints for expressing consistency requirements of a configuration seem to be able to effortlessly capture a significant subset of sales configuration problems. The utility of inheritance in configuration was shown through significant application of the mechanism, especially when related to attributes, and to a lesser but still significant extent to parts.

In addition, this work provides an initial proposal for a framework for characterizing configuration models.

Acknowledgements

We thank A. Anderson, A. Martio, J. Elfström, K. Sartinko, M. Heiskala, M. Pasanen, and T. Kojo for modeling and knowledge acquisition; A. Martio and R. Sulonen for acquisition of the cases; Gardner Denver Finland, KONE, Patria,Tapiola Group for sharing product information; and TEKES for funding WeCoTin, ConSerWe, and Cosmos

References

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.

Damiani, S. R., Brand, T., Sawtelle, M., & Shanzer, H. (2001). Oracle configurator developer User's guide, release 11i Oracle Corporation.

Fleischanderl, G., Friedrich, G., Haselbock, A., Schreiner, H., & Stumptner, M. (1998). Configuring large systems using generative constraint satisfaction. Intelligent Systems and their Applications, IEEE [See also IEEE Intelligent Systems], 13(4), 59-68.

Haag, A. (2008). What makes product configuration viable in a business? Proceedings of ECAI 2008 Workshop on Configuration Systems, Patras, Greece. 53-54.

Kojo, T., Männisto, T. And Soininen, T. (2003). Towards Intelligent Support for Managing Evolution of Configurable Software Product Families. In Software Configuration Management (ICSE Workshops SCM 2001 and SCM 2003 Selected Papers)**,** 86-101.

Nurmilaakso, J. (2004). WeCotin.calc documentation. Unpublished manuscript.

Peltonen, H., Tiihonen, J., & Anderson, A. (2001). Configurator tool concepts and model definition language. Unpublished manuscript.

Schreiber, A. T., & Birmingham, W. P. (1996). Editorial: The Sisyphus-VT initiative. International Journal of Human-Computer Studies, 44(3-4), 275-280.

Soininen, T., Niemelä, I., Tiihonen, J., & Sulonen, R. (2001). Representing configuration knowledge with weight constraint rules. Proceedings of the AAAI Spring Symp.on Answer Set Programming: Towards Efficient and Scalable Knowledge, , 195–201.

Soininen, T., Tiihonen, J., Mannisto, T., & Sulonen, R. (1998). Towards a general ontology of configuration. AI EDAM, 12(04), 357-372.

Tiihonen, J., Heiskala, M., Paloheimo, K., & Anderson, A. (2007). Applying the configuration paradigm to mass-customize contract based services. Paper presented at the Extreme Customization: Proceedings of the MCPC 2007 World Conference on Mass Customization & Personalization, Massachusetts Institute of Technology, MA, USA. paper ID MCPC-134-2007, section 7.5.3.

Tiihonen, J., Soininen, T., Niemelä, I., & Sulonen, R. (2003). A practical tool for mass-customising configurable products. Paper presented at the Proceedings of the 14th International Conference on Engineering Design, Stockholm, Sweden. Paper 1290.

Tiihonen, J., Soininen, T., Niemelä, I., & Sulonen, R. (2002). Empirical testing of a weight constraint rule based configurator. Proceedings of the Configuration Workshop, 15th European Conference on Artificial Intelligence, Lyon, France, 2002. 17–22-17-22.

Wielinga, B., & Schreiber, G. (1997). Configuration-design problem solving. Expert, IEEE [See also IEEE Intelligent Systems and their Applications], 12(2), 49-56.