BRINKOP CONSULTING

K-Modell - Strukturierter Entwurf von Konfigurationsmodellen

Dr. Axel Brinkop

Mit der Methode K-Modell wurde der Notwendigkeit Rechnung getragen, produktspezifische Konfigurationsmodelle systematisch zu entwickeln. Die Methode besteht aus zwei Komponenten: Einer Beschreibungsmethode für Konfigurationswissen und einer Methode zur Erarbeitung des produktspezifischen Konfigurationswissen, bei der die Beschreibungsmethode eingesetzt wird. Sie geht von einer Trennung von Strukturwissen, Wissen über Abhängigkeiten und zu Grunde liegenden Produktdaten aus, um eine effiziente Entwicklung und Pflege des Wissens zu gewährleisten. K-Modell hat sich in einer Vielzahl von Projekten bewährt und als praxisgerecht erwiesen.

1. Die Methode

In der industriellen Praxis versteht man unter einem Produktkonfigurator ein Werkzeug, das hilft ein Produkt gemäß vorgegebenen Anforderungen zu spezifizieren. In der Forschung unterscheidet man feiner zwischen dem Lösen einer Konfigurationsaufgabe, bei der aus einer vorgegebenen Menge von Objekten ausgewählt wird und Parametrisierung, bei der die Eigenschaften des gesuchten (Teil-)Objekts frei spezifiziert werden. (vgl. [Brinkop 1999]). Mit K-Modell können beide Arten von Aufgaben beschrieben werden.

Softwaretechnisch wird ein werkzeugbasierter Produktkonfigurator dadurch realisiert, dass durch eine domänenunabhängige Konfigurationssoftware ein für eine Produktfamilie spezifisches Konfigurationsmodell ausgewertet wird. Das Konfigurationsmodell beinhaltet das Konfigurationswissen.

K-Modell ist eine softwareunabhängige Methode sowohl zur Beschreibung von Konfigurationsmodellen als auch zur Erarbeitung von Konfigurationsmodellen. Sie basiert auf der Annahme, dass Strukturwissen, Wissen über Abhängigkeiten und zu Grunde liegende Daten getrennt repräsentiert werden (vgl. auch [Brinkop 1999; S. 90ff]). Abbildung 1 stellt die Wissensarten dar.

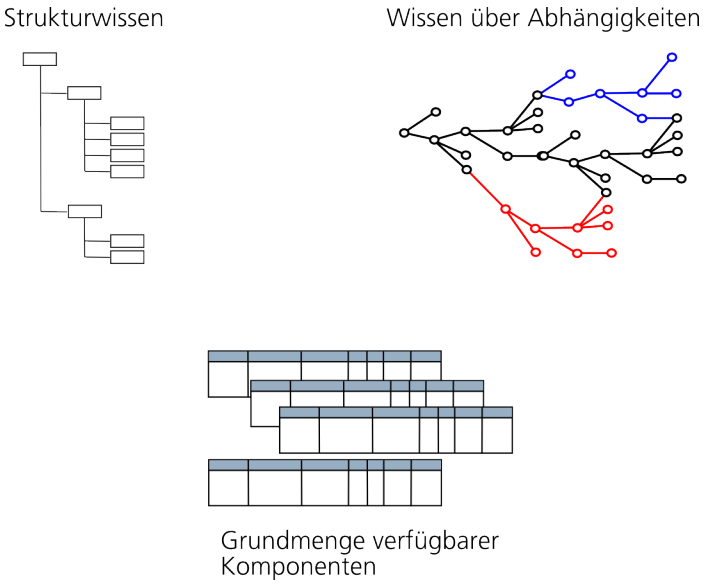


Abbildung 1: Wissensarten

Der Schwerpunkt der K-Modell-Methode liegt auf den Aufgaben, die durch Auswahl aus einer vorgegebenen Grundmenge von Komponenten gelöst werden können. Es können jedoch auch Aufgaben der Parametrisierung damit formuliert werden.

Im Konfigurationsprozess werden durch das Konfigurationswissen die geeigneten Objekte aus der Grundmenge ausgewählt und zu einer Konfiguration mit den gewünschten Eigenschaften kombiniert. Abbildung 2 illustriert die Aufgabenteilung.

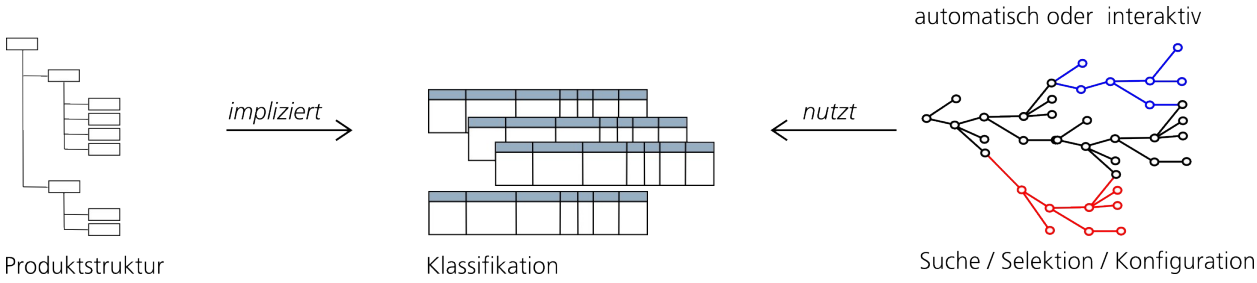


Abbildung 2: Aufgaben der Wissensarten im Konfigurationsprozess

In der Abbildung rechts ist das Wissen über Abhängigkeiten dargestellt. Durch dieses Wissen werden aus den zu Grunde liegenden klassifizierten Produktdaten - in der Mitte dargestellt - die geeigneten Komponenten ausgewählt, entweder voll automatisch oder interaktiv mit dem Nutzer. Auf diese Weise können sowohl Aufgaben der Konfiguration und der Parametrierung als auch der Selektion und Suche gelöst werden.

Die zu Grunde liegenden Komponenten (Baugruppen, Artikel, …) werden in der Klassifikation gepflegt[1]. Gleichartige Komponenten werden auf Grund gleichartiger Merkmale zu Klassen gruppiert, jede Komponente ist durch die konkreten Werte (Ausprägungen) der Merkmale beschrieben.

Im Konfigurationswissen sind die Struktur der Konfigurationsobjekte, die Abhängigkeiten zwischen den Komponenten und Methoden zur Bestimmung von Konfigurationsgrößen hinterlegt.

Durch die unterliegende Konfigurationsengine wird das Konfigurationswisssen mit den Nutzereingaben ausgewertet, um die Anforderungen an die Komponenten zu ermitteln.

K-Modell geht von der Trennung von Konfigurationsmodell und Konfigurationsengine aus. Es werden im Konfigurationsmodell keine Methoden zur Auswertung der Abhängigkeiten formuliert, K-Modell geht davon aus, dass die unterliegende Konfigurationsengine dazu in der Lage ist. Dabei dient kein spezielles Konfigurationswerkszeug als Vorbild. Die Anforderungen sind software-neutral und können durch verschiedene am Markt befindliche Konfigurationswerkzeuge erfüllt werden.

K-Modell ist aus einer Vielzahl von Projekten entstanden. Die Methode wurde für Produktexperten entwickelt, IT-Vorkenntnisse sind nicht notwendig. Sie bildet die Grundlage zur abteilungsübergreifenden Kommunikation über das Vorgehen zur Lösung von Konfigurationsaufgaben.

Die Beschreibungsmethode basiert auf der Darstellung der Zusammenhänge in Form einer Mindmap mit vorgegebener Struktur und fest vorgegebenen Schlüsselwörtern. Die Erstellung der Mindmap erfolgt mit dem OpenSource-Werkzeug Freeplane [Freeplane 2009], dessen Bedienung sehr schnell und einfach zu erlernen ist.

Die Erfahrung zeigt, dass die Methode sehr gut geeignet ist, in abteilungsübergreifenden Workshops mit Teilnehmern aus F&E, Produkt-Management und Vertrieb Konfigurationsmodelle interaktiv zu entwickeln. Durch das Arbeiten mit bekannten Produktzusammenhängen wird der Formalismus von den Workshop-Teilnehmern sehr schnell verstanden und die Diskussionen konzentrieren sich auf das produktspezifische Vorgehen zur Konfiguration. Die Inhalte sind für Fachexperten ohne IT-Vorkenntnisse gut verständlich, auch „Nicht-Techniker" nehmen schnell an der Diskussion teil. Auf dieser Basis können diese abgestimmt definiert und zur Umsetzung in ein Software-Werkzeug freigegeben werden.

1 Klassen können hierarchisch beschrieben werden, die Vererbung von Merkmalen kann ausgedrückt werden.

2. Die Grundmenge

Die Grundmenge der zur Verfügung stehenden Komponenten wird in einer Klassenhierarchie mit Angabe der klassifizierenden Merkmale spezifiziert. Die Hierarchie wird durch das Schlüsselwort `ITEMS` gekennzeichnet.

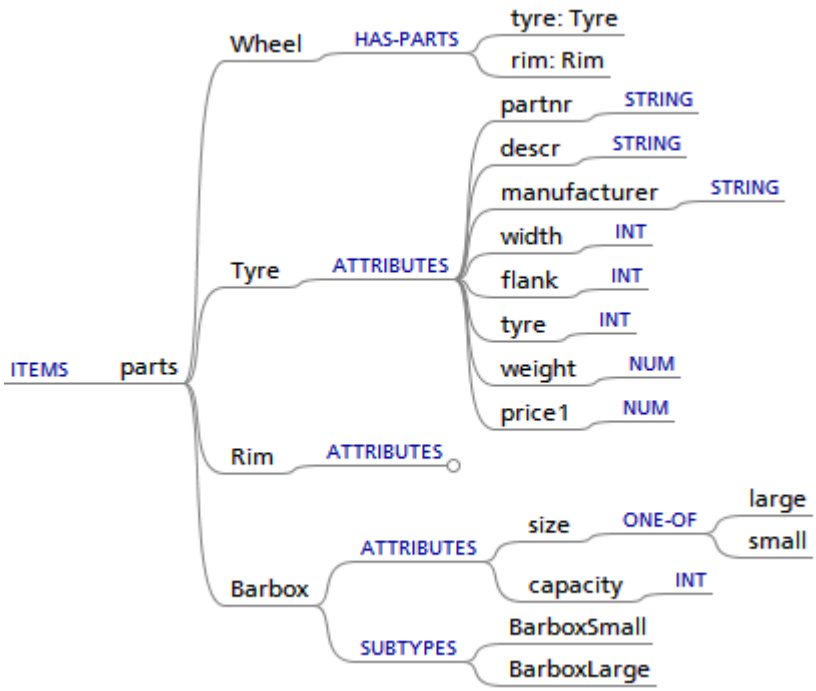


Abbildung 3: `ITEMS` - zu Grunde liegende Klassen

Unter dem Schlüsselwort `ATTRIBUTES` werden die klassifizierenden Merkmale aufgelistet., wobei für jedes Attribut zusätzlich der Datentyp angegeben werden muss (`STRING`, `INT`, `NUM`, `BOOL`) und bei Aufzählungstypen die möglichen Werte nach dem Schlüsselwort `ONE-OF`.

Das Schlüsselwort `SUBTYPES` kennzeichnet die in der Klassenhierarchie untergeordneten Klassen. Attribute werden entlang der Hierarchie vererbt, d.h. in der untergeordneten Klasse sind alle Attribute der übergeordneten Klasse ebenfalls definiert. Für jede Hierarchieebene können damit Attribute hinzukommen, jedoch niemals wegfallen.

Eine Strukturierung in Unterkomponenten lässt sich über das Schlüsselwort `HAS-PARTS` spezifizieren. Darunter können Unterkomponenten in Form von <Rolle> : <Klasse> angegeben werden. Über den Rollennamen kann dann auf die Eigenschaften der Unterkomponente zugegriffen werden.

Durch die Spezifikation im Mindmap werden die Klassen definiert. Die Instanzen der Klassen werden davon getrennt in Tabellen spezifiziert. Diese Tabellen können in Form von Excel-Arbeitsblättern, Datenbank-Tabellen oder CSV-Dateien vorliegen. Die Form hängt von den speziellen Vorlieben der Workshop-Teilnehmer und IT-Rahmenbedingungen ab und können auch später ausgetauscht werden.

3. Die Fragen

Konfigurationsgrößen beschreiben die Anforderungen an das zu konfigurierende Objekt und werden durch das Schlüsselwort `QUESTIONS` eingeleitet. Thematisch zusammengehörende Konfigurationsgrößen werden in Klassen zusammengefasst. Zur besseren Strukturierung können die Attribute einer Klasse zu Themen gruppiert werden.

Dies ergibt eine dreistufige Hierarchie von Klasse – Thema – Attribut, die sich auch in den Bezeichnern von Attributen (= Konfigurationsgrößen) widerspiegelt.

Die Beschränkung auf drei Ebenen ist eine Vereinfachung, die sich in der praktischen Anwendung selten einschränkend bemerkbar macht. Sollten doch einmal mehr Ebenen notwendig sein, lässt sich dies über die Verschachtelung von Klassen darstellen.

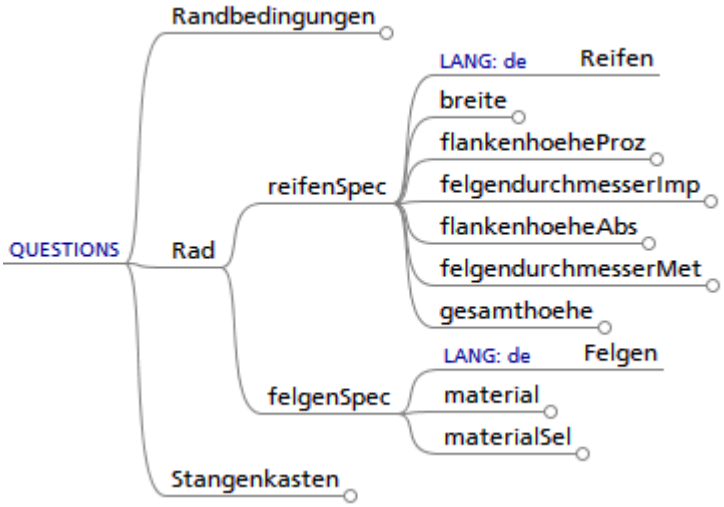


Abbildung 4: `QUESTIONS` - Konfigurationsgrößen

Im Laufe des Konfigurationsprozessses werden die Konfigurationsgrößen entweder interaktiv durch den Nutzer oder durch die Konfigurationsengine bestimmt, um dann zur Auswahl der Komponenten genutzt werden zu können.

Im K-Modell wird vereinfachend davon ausgegangen, dass es keine zusätzliche Ebene zur Definition der Benutzer-Schnittstelle gibt, sondern dass die Konfigurationsgrößen selbst an der Oberfläche repräsentiert sind. Um auszudrücken, dass eine Konfigurationsgröße interaktiv durch den Benutzer bestimmt werden soll, ordnet man der Größe oberflächenrelevante Facetten zu. Zur Verfügung stehen `EDIT`, `SELECT`, `SELEDIT`, `CHECKBOX`.

Von der Konfigurationsengine herzuleitende Größen werden durch das Schlüsselwort `CALC` gekennzeichnet und zusätzlich als `OUTPUT` oder `HIDDEN` spezifiziert. Berechnungen können durch numerische Ausdrücke oder in Form von Entscheidungstabellen formuliert werden. In Abbildung 5 finden Sie Beispiele für die Spezifikation der Oberflächeneigenschaften von Konfigurationsgrößen.

Eine konkrete Implementierung könnte beispielsweise eine Klasse als Register, die Themen als Registerblätter und Konfigurationsgrößen als Ein-Ausgabefelder darstellen.

K-Modell geht von einem sprachunabhängigen Konfigurationsmodell aus, in dem sprachabhängige Darstellungen für Konfigurationsgrößen und deren mögliche Werte spezifiziert werden.

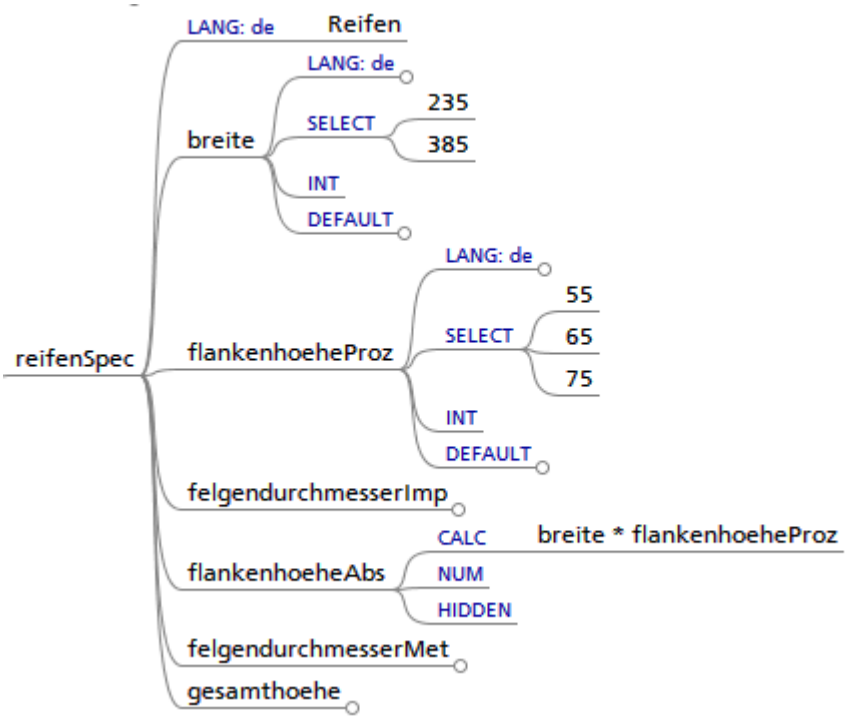
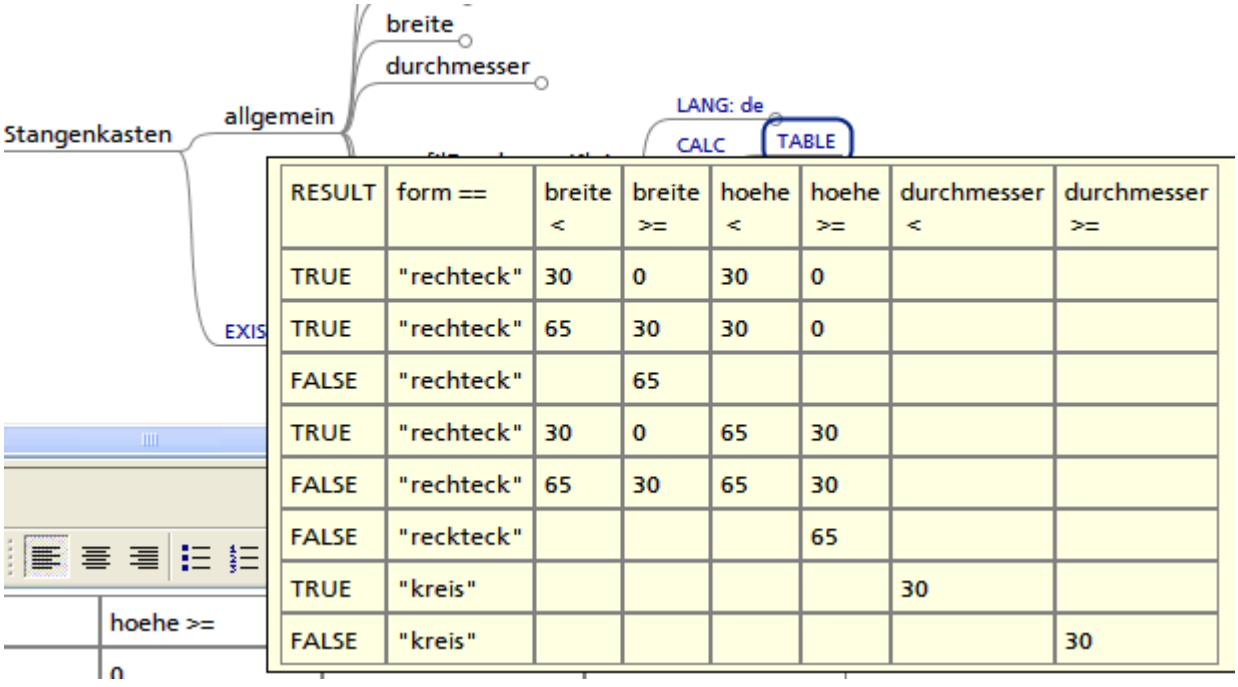


Abbildung 5: Details der Spezifikation von Konfigurationsgrößen

In Abbildung 6 ist ein Beispiel für eine Entscheidungstabelle dargestellt. In der linken Spalte sind die Ergebnisse dargestellt, in allen anderen Spalten Bedingungen. Eine Entscheidungstabelle stellt eine Menge gleich strukturierter Regeln dar, jede Zeile steht für eine Regel. Die Bedingungen der Regeln werden von links nach rechts und von oben nach unten ausgewertet. Die erste Regel, deren Bedingungen alle erfüllt sind, „feuert". Der unter `RESULT` aufgeführte Wert wird zurück geliefert.



RESULT	form ==	breite <	breite >=	hoehe <	hoehe >=	durchmesser <	durchmesser >=
TRUE	"rechteck"	30	0	30	0		
TRUE	"rechteck"	65	30	30	0		
FALSE	"rechteck"		65				
TRUE	"rechteck"	30	0	65	30		
FALSE	"rechteck"	65	30	65	30		
FALSE	"reckteck"				65		
TRUE	"kreis"					30	
FALSE	"kreis"						30

Abbildung 6: TABLE - Berechnung durch Entscheidungstabellen

In K-Modell wird mit statischen und dynamischen Vorgabewerten gearbeitet. Dynamische Vorgabewerte werden analog zu den Berechnungen in Form von Ausdrücken oder Entscheidungstabellen formuliert. Gekennzeichnet werden sie durch das Schlüsselwort `DEFAULT`. Dem Benutzer wird der Wert vorgeschlagen, er kann ihn überschreiben. Vorgabewerte können dynamisch in Abhängigkeit von anderen Konfigurationsgrößen formuliert werden. Ändern sich die Vorbedingungen, so ändern sich auch die Vorgabewerte für Konfigurationsgrößen, die nicht vom Benutzer „berührt" worden sind. Hat der Benutzer bereits den Vorgabewert überschrieben, so bleibt er erhalten.

K-Modell geht davon aus, dass die Fragen in beliebiger Reihenfolge vom Benutzer beantwortet werden können, soweit dies inhaltlich Sinn macht. Inhaltliche Abhängigkeiten zwischen Fragen lassen sich über Existenzbedingungen formulieren, die sich an der Oberfläche als „Sichtbarkeitsregeln" auswirken. In Abbildung 7 ist ein Beispiel dargestellt.

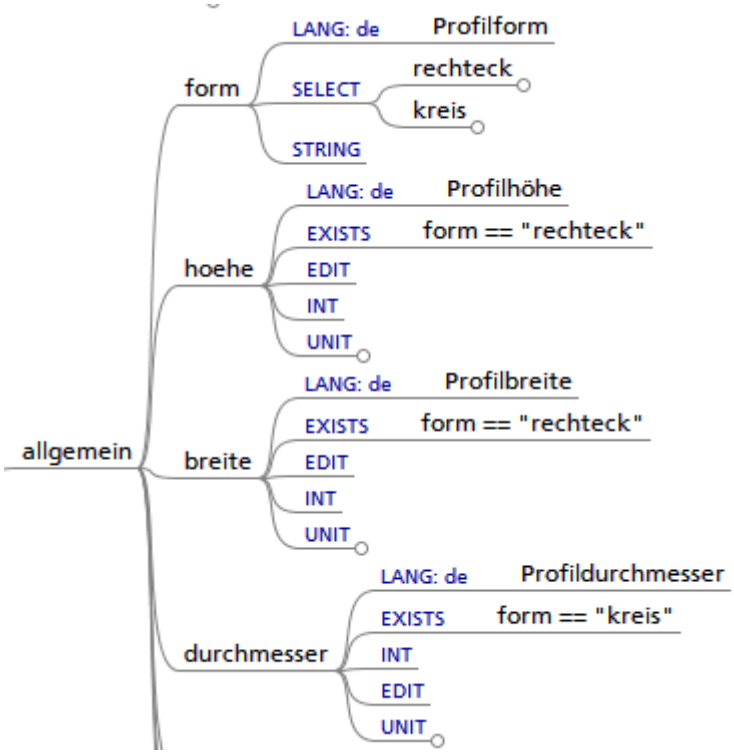


Abbildung 7: `EXISTS` - Existenzbedingungen

Das Beispiel in Abbildung 7 zeigt die Spezifikation einer Profilform. Im Falle eines Rechtecks werden Höhe und Breite abgefragt, im Fall eines Kreises, der Durchmesser.

Durch das Schlüsselwort `MANDATORY` können Eingaben als Pflichtfelder definiert werden. Die Information, ob alle Pflichtfelder ausgefüllt wurden kann man in der Konfiguration nachgelagerten Prozessschritten wie der Angebotserstellung nutzen.

4. Das Ergebnis

Das Ergebnis des Konfigurationsprozesses ist eine Vertriebsstückliste, die im K-Modell-Mindmap mit `BOM` gekennzeichnet ist. Die Stückliste kann beliebig tief strukturiert sein. Die Positionen werden üblicherweise durch Objekte der Grundmenge besetzt.

Die Zuordnung erfolgt in Form von <Rolle> : <Klasse>, dadurch wird der Position mit dem Bezeichner <Rolle> eine Instanz der Klasse <Klasse> zugeordnet. Durch Einschränkung der Attribute der Klasse wird die Zuordnung der Instanz definiert. Abbildung 8 soll dies verdeutlichen.

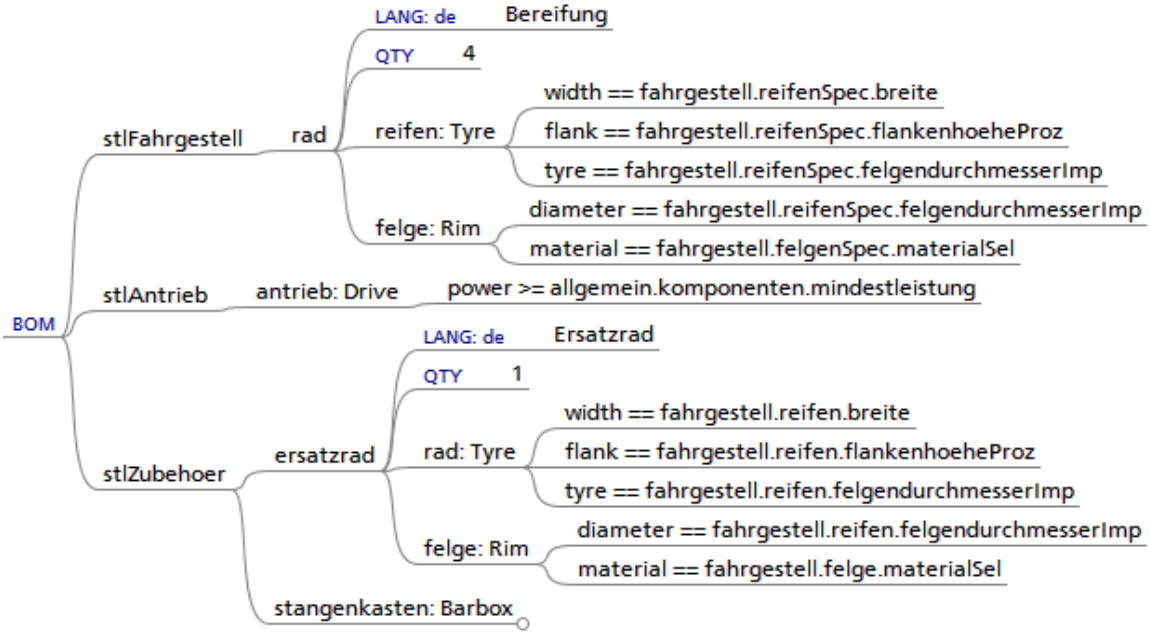


Abbildung 8: `BOM` - Definition der Stückliste

Durch diese Spezifikation wird aus der Grundmenge der Klasse der Motoren `Drive`, diejenige Instanz ausgewählt, deren Leistung `power` den durch `mindestleistung` spezifizierten Anforderungen genügt.

Die Klasse der Motoren muss mit all ihren Attributen unter `ITEMS` deklariert worden sein. Die Instanzen sind davon getrennt in den Grunddaten definiert. Durch die Zuordnung einer Instanz zu einer Stücklistenposition werden alle definierten Attribute durch die Attributswerte der ausgewählten Instanz mit Werten belegt.

Dieses Verhalten kann man ergänzen oder überschreiben, indem man eine Berechnungsvorschrift an einem Attribut mittels `CALC` hinterlegt. Dies kann man nutzen, um dynamisch Merkmale aufgrund der aktuellen Konfiguration zu setzen.

Durch das Schlüsselwort `QTY` wird die Menge der einzusetzenden Artikeln spezifiziert. Ohne diese Angabe wird die Menge 1 angenommen. Zur Spezifikation können beliebige Ausdrücke analog zu `CALC` verwendet werden.

Enthält eine Klasse Komponenten, die über `HAS-PARTS` deklariert sind, so muss für jede dieser Unterkomponenten deren Einschränkungen angegeben werden.

Die Stücklistenposition wird nur bei einer eindeutigen Zuordnung durch die Instanz der Grundmenge (= Artikel) ersetzt. Erfüllt keiner oder mehr als ein Artikel die Anforderungen so bleibt sie unspezifiziert, was nicht gewünscht ist.

Im obigen Beispiel würden alle Motoren mit einer größeren Leistung als spezifiziert ausgewählt. Würden die Grunddaten mehr als einen Motor mit der geforderten Mindestleistung enthalten, würde die Stücklistenposition unspezifiziert bleiben. Um den von der Leistung kleinsten Motor auszuwählen, der den Anforderungen genügt, kann durch `ORDER-BY` eine Sortierung in Kombination mit `ASC` und `DESC` angegeben werden. Durch das Schlüsselwort `FIRST` wird das erste Element der Sortierung ausgewählt.

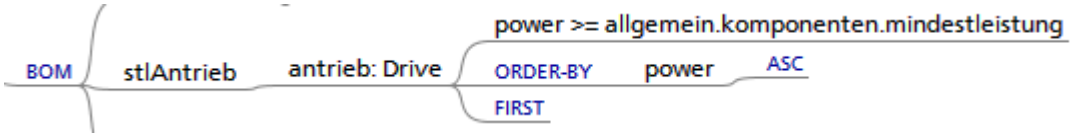


Abbildung 9: Sortieren mit ORDER-BY

Soll eine Stücklistenposition nur in bestimmten Situationen erscheinen, so kann sie mit einer Existenzbedingung versehen werden. Nur wenn die Bedingung erfüllt ist, erscheint die Position in der Stückliste. Dies kann auf allen Stücklistenebenen genutzt werden.

Für Zwischenebenen kann man Strukturknoten verwenden, für die es keine Entsprechung in den zu Grunde liegenden Daten gibt. Über die bereits vorgestellten Schlüsselwörter CALC und LANG: de wird einem Strukturknoten eine Artikelnummer und eine sprachspezifische Bezeichnung zugeordnet.

In Abbildung 10 ist die Stückliste mit den Erweiterungen wiedergegeben.

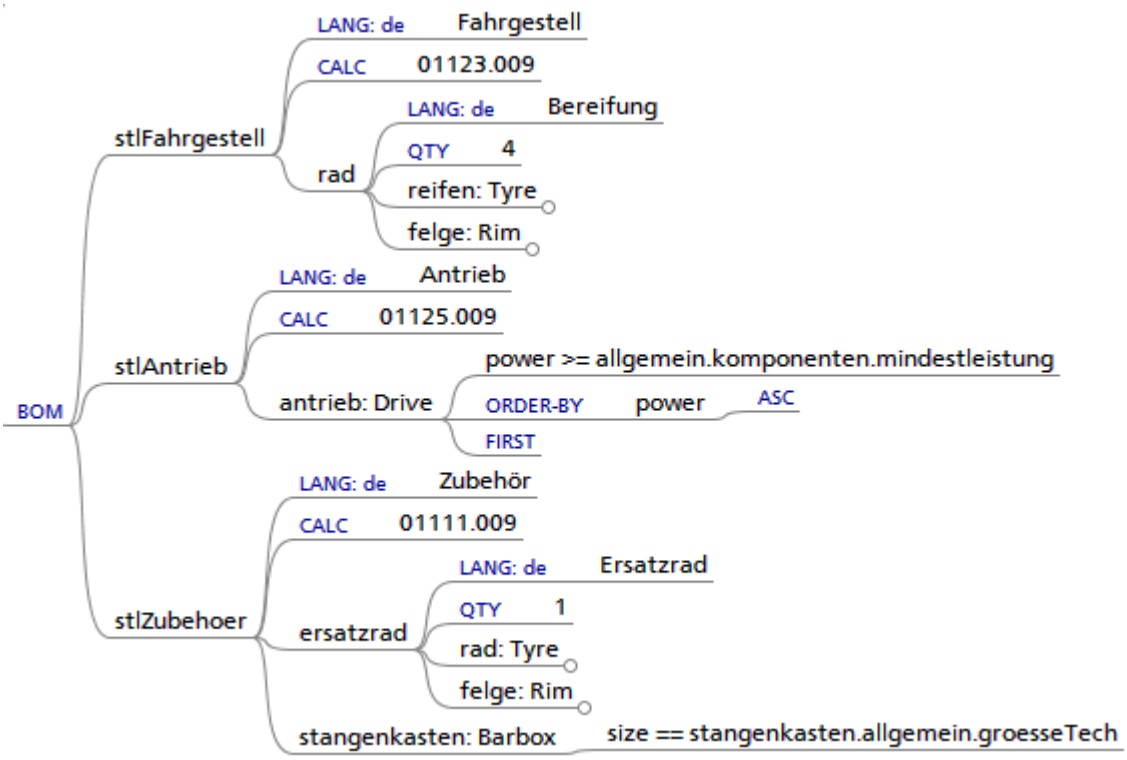


Abbildung 10: Stückliste mit Strukturknoten

5. Der Katalog

Der Katalog ist der Startpunkt des Benutzers. Dort sucht der Benutzer nach geeigneten konfigurierbaren und voll spezifizierten Produkten. Der Katalog ist nach Kategorien strukturiert. Jede Kategorie kann weitere Unterkategorien oder Produkte enthalten. Ein Produkt kann in mehreren Kategorien enthalten sein, d.h. die Zuordnung Produkt – Kategorie ist nicht eindeutig. Der Benutzer kann ein Produkt auf mehreren Wegen finden. Abbildung 11 gibt ein Beispiel.

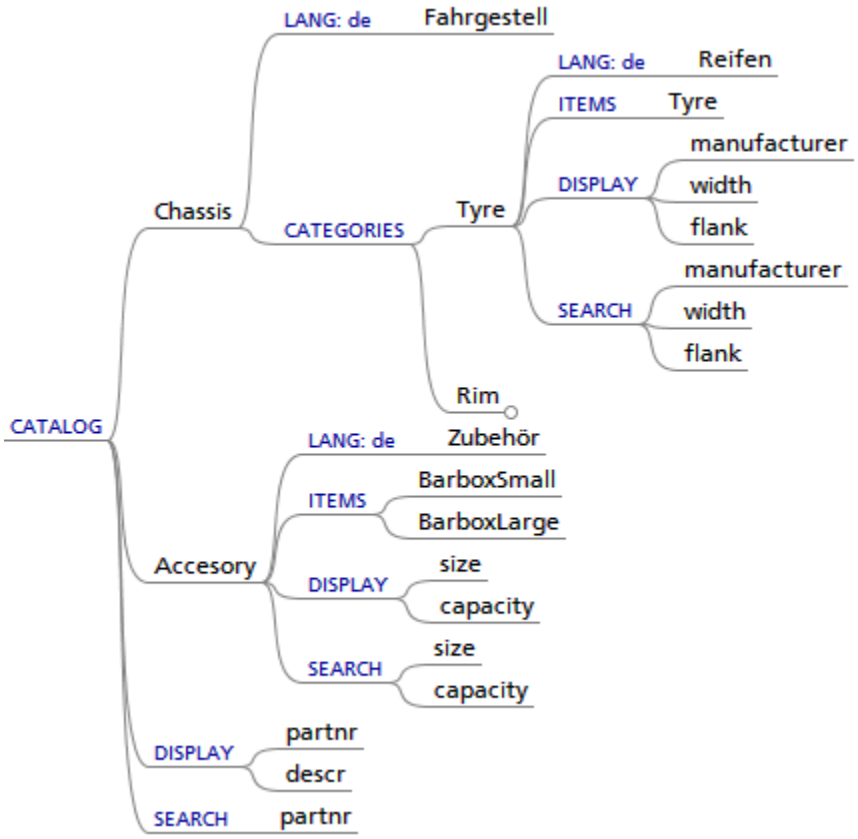


Abbildung 11: `CATALOG` - Definition des Produktkatalogs

In diesem Beispiel werden die Inhalte der Kategorien entweder durch die Angabe von Unterkategorien (`CATEGORIES`) oder durch die Angabe von Klassen (`ITEMS`) spezifiziert. Alle Instanzen der angegebenen Klasse werden dann dieser Kategorie zugeordnet. Weiterhin ist es möglich konfigurierbare Produkte durch das Schlüsselwort `CONF` anzugeben.

Zu jedem dargestellten Produkt / Artikel werden die mittels `DISPLAY` spezifizierten Attribute im Katalog dargestellt und können vom Benutzer als Auswahlkriterium genutzt werden. Über die mittels `SEARCH` spezifizierten Attribute kann der Benutzer gezielt nach bestimmten Eigenschaften suchen oder filtern. Es gelten hier Vererbungsmechanismen, die auf einer höheren Ebene für `DISPLAY` und `SEARCH` spezifizierten Attribute sind auch für die aktuelle Ebene verfügbar.

A.B.

6. Der Modellierungsprozess

6.1 Konfigurationsgrößen erfassen

Konfigurationsgrößen werden am besten im Team in Workshops mit Brainstorming-Charakter erfasst. Bei derartigen Workshops geht es nicht darum, das Konfigurationsmodell im Detail festzulegen, sondern die Thematik in der Breite zu erfassen. Daher eignet sich eine informelle, mitlaufende Protokollierung in Form von Mindmaps sehr gut.

Folgende Sachverhalte werden für jede Konfigurationsgröße festgehalten.

- handelt es sich um eine Eingabegröße?
    - wird der Wert frei eingegeben?
    - wird der Wert aus einer Liste ausgewählt?
- handelt es sich um eine Ausgabegröße?

6.2 Konfigurationsgrößen strukturieren

Thematisch zusammengehörende Konfigurationsgrößen werden zu Klassen gruppiert. Um die Handhabbarkeit einer Klasse auch bei einer großen Zahl von Konfigurationsgrößen zu gewährleisten, bietet K-Modell die Möglichkeit, die Konfigurationsgrößen innerhalb einer Klasse mittels sogenannten „Themen" feiner zu strukturieren.

K-Modell geht davon aus, dass die Konfigurationsgrößen ohne zusätzliche Präsentationsschicht unmittelbar an der Oberfläche präsentiert werden. Die Strukturierung in Themen kann als eine Präsentation der Konfigurationsgrößen auf verschiedenen Registerblättern interpretiert werden, die Konfigurationsgrößen eines Themas werden dem Benutzer „gleichzeitig" in der im K-Modell definierten Reihenfolge präsentiert. Wird das Thema gewechselt, so werden die Konfigurationsgrößen des neuen Themas dargestellt, alle anderen sind „ausgeblendet". Aus diesem Grund ist es wichtig, sich gegenseitig stark beeinflussende Konfigurationsgrößen in einem Thema zu platzieren.

6.3 Verfügbare Komponenten strukturieren

Bei der Strukturierung der verfügbaren Komponenten geht man von dem gewünschten Ergebnis der Konfiguration aus. Handelt es sich beispielsweise um eine vertriebliche Konfiguration, so ist das gewünschte Ergebnisse eine Vertriebsstückliste. Bei den zugrunde liegenden Komponenten handelt es sich um Vertriebspositionen.

Will man Kostentransparenz erzielen, muss eine eindeutige Zuordnung der Vertriebsstückliste zu Postionen der Fertigungsstückliste gewährleistet sein. In diesem Fall dient die technische Produktstruktur als Basis zur Strukturierung der verfügbaren Komponenten.

Grundsätzlich werden die verfügbaren Komponenten in Klassen mit Attributen eingeteilt. In einer Abbildung auf ein ERP-System entspricht das Vorgehen einer Klassifizierung der Stammdaten mit einer Merkmalsbewertung.

Im Konfigurationsprozess sollen merkmalsbasiert die geeigneten Komponenten gewählt werden können. Die Attribute repräsentieren die für diesen Zweck charakteristischen Eigenschaften, sie sollen die Komponenten hinsichtlich ihre Eignung unterscheiden. Sollen die Komponenten automatisiert ausgewählt werden, dann muss es möglich sein, immer genau eine Komponente zu finden, mehrfache Treffer dürfen nicht auftreten. Dies bedeutet, dass es keine zwei Komponenten mit identischen Merkmalswerten geben darf. Ist dies doch der Fall, so muss ein weiteres Attribut eingeführt werden, das es erlaubt, zwischen ihnen zu unterscheiden.

Komponenten mit identischen Merkmalswerten sind jedoch erlaubt, wenn die Komponenten interaktiv durch den Benutzer ausgewählt werden sollen. Dann muss die Information über die Unterschiede der Komponenten in beschreibenden Text oder in grafische Darstellungen untergebracht werden, die dem Benutzer angezeigt werden.

6.4 Abhängigkeiten identifizieren

Abhängigkeiten zwischen den Konfigurationsgrößen bestimmen den Schwierigkeitsgrad der Lösung eines Konfigurationsproblems. Die Größen können sich auf verschiedene Arten gegenseitig beeinflussen:

- Berechnungen
  Der Wert einer Konfigurationsgröße ist abhängig von anderen Konfigurationsgrößen, die Berechnung ist beispielsweise als Formel mit Variablen definiert.
- Selektionskriterien
  Komponenten werden aufgrund von Werten der Konfigurationsgrößen ausgewählt. Ein typisches Beispiel ist die oben erwähnte Auswahl eines Motors mit vorgegebener Mindestleistung.
- Vorgabewerte
  Vorgabewerte können dynamisch in Abhängigkeit von anderen Konfigurationsgrößen formuliert werden.
- Existenzbedingungen
  Konfigurationsgrößen existieren nur unter bestimmten Bedingungen, beispielsweise existiert in Abbildung 7 der Durchmesser nur für ein rundes Profil.

Diese Abhängigkeiten werden zuerst informell erfasst. Hilfreich ist die Erstellung einer Abhängigkeitsmatrix in der die Art der Abhängigkeit eingetragen wird.

6.5 Abhängigkeiten analysieren

Generell gilt, dass stark voneinander abhängige Konfigurationsgrößen auch thematisch zusammen gehören. Andererseits soll die Zahl der Attribute einer Klasse eine bestimmte Grenze nicht überschreiten. Die Analyse der Abhängigkeiten gibt Hinweise darauf, welche Güte die gewählte Strukturierung der Konfigurationsgrößen hat.

6.6 Abhängigkeiten formalisieren

Schließlich werden die Abhängigkeiten formal in Form von Formeln, Algorithmen und Auswahl-Statements aufgeschrieben. Dieser Schritt ist notwendig, um die Umsetzbarkeit in ein Software-Werkzeug zu gewährleisten.

Wichtig ist dabei, die informelle Beschreibung der Abhängigkeit parallel beizubehalten, um einerseits eine Kontrollmöglichkeit zu haben und andererseits die Kommunikation zu erleichtern.

7. Zusammenfassung

K-Modell ist eine softwareunabhängige Methode zur Entwicklung und zur Beschreibung von Konfigurationsmodellen in Form von speziell strukturierten Mindmaps. Sie basiert auf dem Ansatz, Strukturwissen, Wissen über Abhängigkeiten und Grunddaten getrennt zu beschreiben.

Die Methode zielt auf die Kommunikation des Konfigurationswissens zwischen den verschiedenen Fachabteilungen, von F&E über Produktmanagement, Vertrieb und Fertigung bis zur Geschäftsleitung, um die Inhalte diskutieren und festlegen zu können.

Typischerweise ist das Konfigurationswissen im Unternehmen auf eine Vielzahl von Köpfen verteilt. Mit K-Modell wird eine strukturierte Modellierungsmethode eingesetzt, bei der in Workshops das Konfigurationsmodell gemeinsam systematisch erarbeitet wird. Diese Vorgehensweise hat sich in einer Vielzahl von industriellen Projekten als effiziente Herangehensweise an Konfigurationsprobleme erwiesen.

8. Literatur

[Brinkop 1999] Axel Brinkop: "Variantenkonstruktion durch Auswertung der Abhängigkeiten zwischen den Konstruktionsbauteilen", Dissertationen zur Künstlichen Intelligenz, Band 204, Infix, St.-Augustin, 1999

[Freeplane 2009] http://freeplane.sourceforge.net