*Variantenmanagement in AUTOSAR*

# »#ifdef« in XML?

**Bisher handhabten Softwareentwickler das Management von Varianten »irgendwie«, meist mit parametrierten Makefiles und dem C-Preprozessor. Bei der Überführung bestehender Software in AUTOSAR-konforme Strukturen tritt Variantenmanagement als Problem in den Vordergrund und verlangt nach einer Lösung. Denn »#ifdef« und Co. sind in AUTOSAR nicht mehr nutzbar.**

**Mario Friedrich**
**Dr. Jörg-Volker Müller**

Softwarezulieferer im Automobilbereich haben die Aufgabe, Varianten einer existierenden Softwarefamilie (Softwareproduktlinien) möglichst schnell zur Verfügung zu stellen. Dabei muss eine Variante Funktionen bereitstellen, die von Anforderungen wie generellen Vorgaben des OEM, dem jeweiligen (eventuell regional unterschiedlichen) Fahrzeugmodell und dem Stand des Software-Lebenszyklus beeinflusst werden.

Um eine Softwarevariante zu erstellen, ist es notwendig, einzelne Module zu ersetzen, zu modifizieren und/oder zu konfigurieren. Zur Lösung des Variantenmanagement-Problems auf technischer Ebene kommt bisher verstärkt der C-Preprozessor zum Einsatz. Er allein reicht jedoch nicht aus, und so greifen die Softwareentwickler zusätzlich auf parametrierte Makefiles (grobe Vorauswahl von Modulen durch den »Build«-Prozess) und Konfigurationstools (Erzeugen von Konfigurationsdateien) zurück. Jedes dieser Hilfsmittel löst jedoch nur einen Teil der Aufgabe. Zudem erschweren das Aufteilen der Aufgabe auf ganz unterschiedliche Werkzeuge und deren Verteilung im Softwareentwicklungsprozess das Variantenmanagement zusätzlich. Im ungünstigsten Fall kommt es zu »Konkurrenzsituationen« der angewandten Methoden und somit zu Fehlern, deren Ursache schwer zu analysieren ist.

Bild 1 veranschaulicht, wie bisher Softwarevarianten erstellt wurden. In einer Datei wie `Features.h` legt der

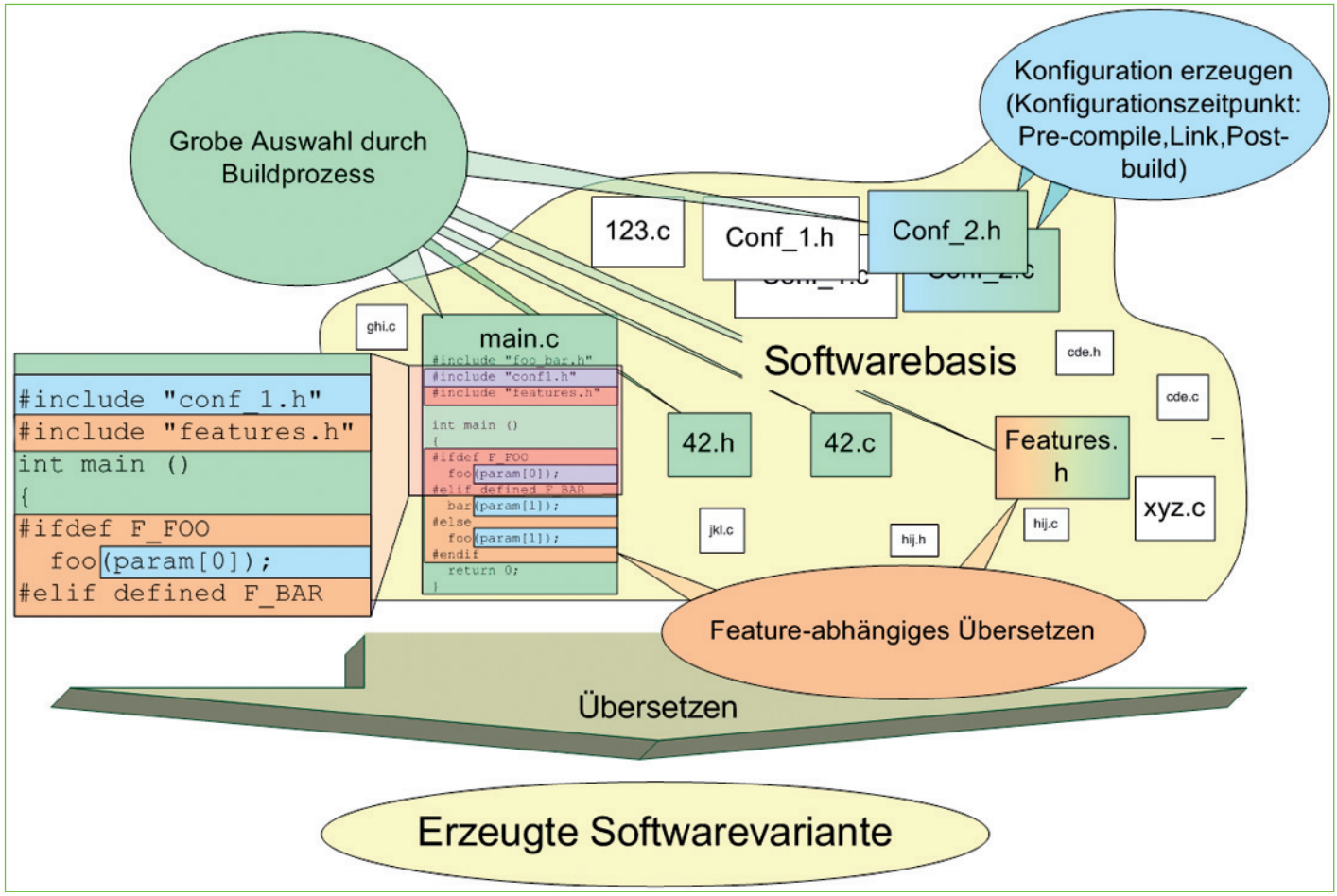


**Bild 1: Bisher erfolgte das Variantenmanagement über bedingte Kompilierung, einem Zusammenspiel aus parametrierten Makefiles und dem C-Preprozessor**

Entwickler die gewünschten Features (Eigenschaften) durch »`define`«-Statements fest. Dies sieht dann üblicherweise so aus: `#define FEATURE_FOO`. Ist dieses Feature nicht mehr gewünscht, wird diese Zeile einfach auskommentiert: `//#define FEATURE_FOO`. Ein »`define`«-Statement dient somit als Schalter, über den sich bestimmte Features mittels einer Konfigurationsdatei auswählen lassen. Im übrigen Quelltext kommt dann ein »`#ifdef ... #endif`«-Ausdruck zum Einsatz, um abhängig von diesen Schaltern die benötigten Teile der Software zu übersetzen. Dies wird auch als »Conditional Compilation« bezeichnet. Da selbst in kleinen bis mittleren Projekten (ab zwei Mannjahre Implementierungszeit) 300 »`define`«-Statements schnell erreicht sind, führt dieser Ansatz frühzeitig zu Problemen der Handhabbarkeit.

»Conditional Compilation« (bedingte Kompilierung) ist ein seit langem intensiv genutztes technisches Hilfsmittel. Wichtig ist die Erkenntnis, dass es sich hierbei wirklich nur um ein technisches Hilfsmittel handelt und dieses Verfahren nicht als Variantenmanagement bezeichnet werden kann. Einige der größten Probleme, die hierdurch entstehen, sind:

- Das Variantenmanagement ist im Quellcode »vergraben«,
- Abhängigkeiten zwischen verschiedenen Features sind auf Grund der eingeschränkten Syntax des C-Preprozessors nur schwer zu handhaben, denn er muss über eine ganze Kette von Abhängigkeiten analysieren, ob sich Module gegenseitig bedingen oder ausschließen,
- durch verschachtelte Abhängigkeiten wird der Code schwer lesbar,
- bedingte Kompilierung funktioniert nicht mit AUTOSAR.

Das Zusammenspiel zwischen Conditional-Compilation und AUTOSAR klappt nicht, weil entsprechend den AUTOSAR-Vorgaben Teile der Softwarelogik in XML beschrieben werden (vornehmlich die Kommunikationsverbindungen zwischen den Modulen) und XML keine Abhängigkeiten kennt. Auf den Punkt gebracht: Es gibt kein `#ifdef` in XML.

## *Varianten mit speziellen Tools managen*

Spätestens in Verbindung mit AUTOSAR muss also eine »richtige« Variantenmanagement-Lösung zum Einsatz kommen, welche die Aufgabe umfassend und somit auf einer höheren Ebene als dem Quelltext lösen kann. Grundsätzlich ist die Entwicklung von AUTOSAR-konformer Software immer eng mit dem Einsatz von Toolketten verbunden. Die einzelnen Werkzeuge lesen Quellcode und XML-Dateien ein, bereiten die Daten in einer grafischen Entwicklungsumgebung zur Bearbeitung auf und geben dann wieder XML und Quellcode aus. So lassen sich sehr flexibel individuelle Toolketten aufbauen und in einem Projekt einsetzen. Ein entscheidender Bestandteil einer derartigen AUTOSAR-Toolkette wird zukünftig das Variantenmanagementsystem sein. Es hat folgende Vorteile gegenüber den bisherigen Mitteln:

- Es macht das Management von Varianten explizit, das heißt, es ist nicht mehr im Quelltext verborgen,
- es stellt Abhängigkeiten grafisch dar, schlägt mögliche Features vor und zeigt Konflikte auf,
- große Systeme lassen sich durch eine hierarchische

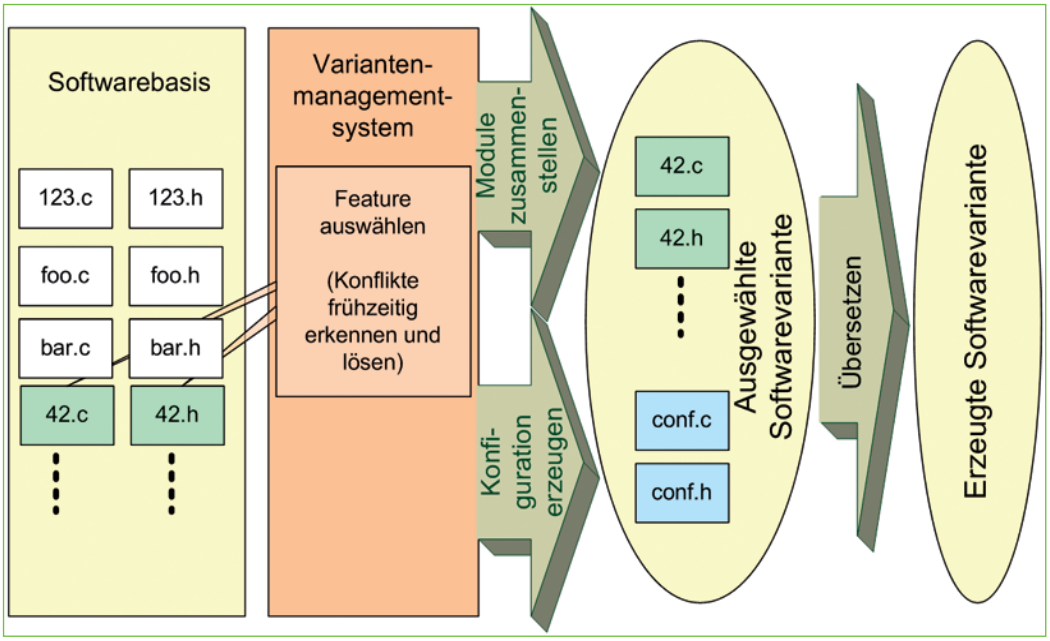


**Bild 2: In einer AUTOSAR-Umgebung der Zukunft erlauben spezielle Tools einen strukturierten Ansatz für das Variantenmanagement**

Aufteilung in Layer und Module handhaben,

- die Variantenauswahl erfolgt in einer grafischen Oberfläche und nicht in einer im Quelltext vergrabenen Feature-Datei,
- Variantenmanagementsysteme sind meist sehr eng mit Requirementmanagementsystemen verknüpft, so lassen sich Anforderungen bidirektional bis zur erstellten Variante verfolgen,
- Dokumentationen zur gewählten Softwarevariante im Microsoft-Word-Format können selbst generiert werden und
- Variantenmanagement funktioniert sehr gut mit AUTOSAR, denn es erzeugt und modifiziert jede Art von Code wie Makefiles, C, C++, XML usw. und importiert und exportiert Systembeschreibungen und Konfigurationsdaten üblicherweise im XML-Format, wie in AUTOSAR vorgesehen.

Bild 2 veranschaulicht die Funktionsweise eines Variantenmanagementsystems und zeigt deutlich die klare Strukturierung mit genau einem System, das aus einer Basis von Softwareelementen auswählt – eine klare Verbesserung zum bisherigen Vorgehen mit verschiedenen Mechanismen (Bild 1). Zunächst werden im System die Features ausgewählt, die die spätere Variante einmal haben soll. Abhängig davon wählt das System dann »Elemente« aus einer bestehenden Softwarebasis aus, und der letzte Schritt ist die Generierung des Systems mit den gewünschten Eigenschaften.

Elemente der Softwarebasis können dabei ganze Dateien (`*.c *.cpp *.obj *.xml *.doc`) oder auch nur einzelne Zeilen Quellcode sein. Der Generierungsschritt lässt sich dabei flexible den eigenen Anforderungen anpassen, dadurch wird ein breites Einsatzspektrum sichergestellt. Es ist auch möglich, bestehenden Code zu modifizieren (ähnlich wie es der C-Preprozessor macht).

## Grafik macht Varianten sichtbar

Ein kleines Beispiel, in dem Variantenmanagement auf zwei AUTOSAR-Softwarekomponten angewendet wird, soll der Verdeutlichung dienen. AUTOSAR-Tools bieten meist eine grafische Entwicklungsumgebung. Die Software wird hier auf VFB-Ebene (Virtual Functional Bus) modelliert und konfiguriert. Das folgende Beispiel (siehe Bild 3) zeigt die beiden Softwarekomponenten `InstrumentCluster` und `doorStatus`. Dabei ist `InstrumentCluster` eine Anwendungs-Softwarekomponente und `doorStatus` eine Sensor-Softwarekomponente. Abhängig vom Feature `Comfort` soll `Ins-`

| | C-Preprozessor | XML |
|---|---|---|
| Codefragment | `#define CIN4711 doorStatus()` | `<receive port="doorStatus" />` |
| Variantenabhängig | `#ifdef (FEATURE_Comfort)`<br>`# define CIN4711 doorStatus()`<br>`#endif` | `<receive`<br>`condition=`<br>`"pv:hasFeature ('Comfort')`<br>`"port="doorStatus"`<br>`/>` |
| Ergebnis, wenn das Feature `Comfort` ausgewählt ist | `#define CIN4711 doorStatus()` | `<receive port="doorStatus"/>` |
| Ergebnis, wenn das Feature `Comfort` nicht ausgewählt ist | **nichts** | **nichts** |

**Tabelle 1: Gegenüberstellung C-Preprozessor und ein featureabhängiger AUTOSAR-XML-Ausschnitt**

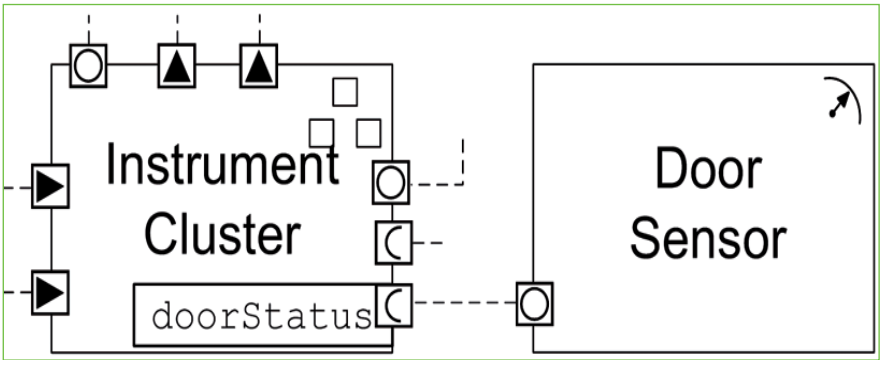


**Bild 3: Beispiel für eine Softwarevariante: In der Komponente `doorStatus` wird der `Door Sensor` ausgelesen**

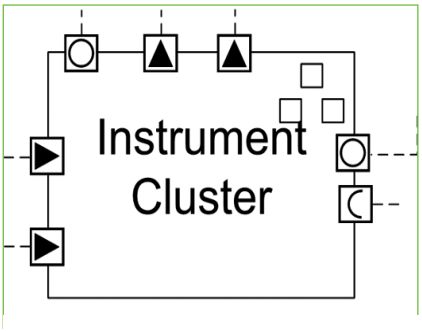


**Bild 4: Ohne das Software-Feature `Comfort` existieren weder `Door Sensor` noch `doorStatus`**

`trumentCluster` den Port `doorStatus` besitzen und über diesen Port mit dem `DoorSensor` verbunden sein. Ist das Feature `Comfort` nicht ausgewählt, so existiert der Port `doorStatus` nicht. Unter der Voraussetzung, dass es auch sonst keine Softwarekomponenten gibt, die von der Komponente `DoorSensor` abhängig sind, gibt es diese in der erzeugten Variante überhaupt nicht (siehe Bild 4).

Tabelle 1 zeigt der Vollständigkeit halber, wie ein `#ifdef` in XML umgesetzt wird, dabei kam in der XML-Variante die Notation des Variantenmanagementsystems »pure::variants« [1] zum Einsatz. Zukünftig wird eine Betrachtung auf dieser Ebene aber nicht mehr nötig sein.

AUTOSAR deckt jedoch nur einen bestimmten Bereich der Softwareentwicklung ab, deshalb muss weiterhin sichergestellt sein, dass Variantenmanagement auch in den übergeordneten Software-Entwicklungsprozess passt. Variantenmanagement fügt sich nicht nur ideal in die AUTOSAR-Welt ein, sondern unterstützt den heutigen durch »SPICE« und »CMMI« beeinflussten Software-Entwicklungsprozess sehr gut. So lässt sich die Machbarkeit sehr früh anhand von gesammelten Anforderungen (Requirements) und einer ersten »groben« Modellierung abschätzen, Konflikte oder fehlende Funktionen werden schnell erkannt. Da es entweder eine direkte Kopplung zwischen dem Requirement-Managementsystem und dem Varianten-Managementsystem gibt oder die Anforderungen direkt im Variantenmanagementsystem verwaltet werden, ist ein bidirektionales Mapping von den Requirements zum Arbeitsprodukt jederzeit möglich. Somit spiegelt das entwickelte Softwareprodukt direkt den Stand der Anforderungen wider. Um die andere Seite des V-Modells des Software-Entwicklungsprozesses zu unterstützen, stellt das Variantenmanagementsystem die für die erzeugte Variante relevanten Testfälle bereit. Kürzlich erschienene Beiträge ([2] und [3]) zeigen, dass Variantenmanagement nicht nur im Automobilsektor ein aktuelles Thema ist. (cg)

**Mario Friedrich**
ist AUTOSAR-Consultant,
**Dr. Jörg-Volker Müller**
ist technischer Leiter, beide bei
**Lineas Automotive**
**Telefon 05 31/88 52 0**
**www.lineas.de**

**Literatur**

[1] www.pure-systems.com, Varianten-management-Dokumentation, 2008.
[2] Dziobek, C.; Loew, J.; Prystas, W.; Weiland, J.: Von Vielfalt und Variabilität, Elektronik automotive, WEKA FACHMEDIEN GmbH, 03.2008.
[3] Prof. Dr. Schmidt, K.: Gleichheit in Vielfalt, iX, Heise Zeitschriften Verlag, 05.2008