

# Modellgetriebene Steuergeräte- Entwicklungsumgebung für Produktlinien

Dr. Lars Geyer-Blaumeiser, Robert Bosch GmbH  
Dr. Lothar Wendehals, itemis AG

OOP 2009, München  
27.01.2009

**itemis**  
Software | Consulting | Coaching



## Motivation

### Ziel

- Effizienzverbesserung der Steuergeräte-Softwareentwicklung
- Qualitativ hochwertige Steuergeräte-Software durch bestmögliche Unterstützung des Entwicklers
- Integrierte Werkzeugumgebung zur Entwicklung und Dokumentation der Steuergeräte-Software

## Agenda

- Motivation
- Der Steuergeräte-Software-Entwicklungsprozess
- Technische Grundlagen
- Unterstützung des Entwicklers
- Fragen und Diskussion

## Steuergeräte-Software

- Eingebettetes System
  - strikte Hardware-Limitierungen, hoher Kostendruck
- Programmiersprache C, Ganzzahlarithmetik
- Metadaten wie Kennlinien/-felder zur Kalibrierung der Software
- Basiert auf **Komponentenmodell**
- Software aufgeteilt auf **Komponenten (Dekomposition)**
  - Struktur- und Funktionskomponenten
  - Strukturkomponenten enthalten wiederum Komponenten
  - Funktionskomponenten enthalten C-Funktionen
- Unterschiedlichste Zielumgebungen (Autos verschiedenster Hersteller)
- Aber Wiederverwendung der Komponenten → **Produktlinien**

## Produktlinienentwicklung

- Eine Produktlinie für die gesamte, aktuelle Steuergerätegeneration
- Produktlinie enthält zahlreiche Varianten für die verschiedensten Zielplattformen
- **Variantenbildung** durch
  - Austausch von kompletten Komponenten
  - Austausch von Teilkomponenten einer Dekomposition (Struktur- oder Funktionskomponenten)
  - Präprozessor-Anweisungen in den C-Funktionen einer Komponente
- **Konfiguration** von Komponenten, Funktionen und Variablen definiert konkretes Produkt

## Aktuelles Werkzeug-Ökosystem

### **Werkzeuge in unterschiedlichen Technologien für**

- Software Konfigurationsmanagement
- die Modellierung von Signalflüssen, Steuerungen und Regelungen
- für die Bearbeitung der Software-Komponenten und der Dokumentation
- C-Editoren zur Funktionsentwicklung
- die Codegenerierung und das Software-Build-System

### **Verbesserungspotential:**

- domänenspezifische Unterstützung
- Optimierung der Datenhaltung
- Reduktion des Wartungsaufwands

### **Randbedingungen:**

- Sicherheitskritische Anwendungen
- Lange Verwendung (Änderungen bis zu 10 J. nach Auslieferung)

### Vorteile einer integrierten Entwicklungsumgebung

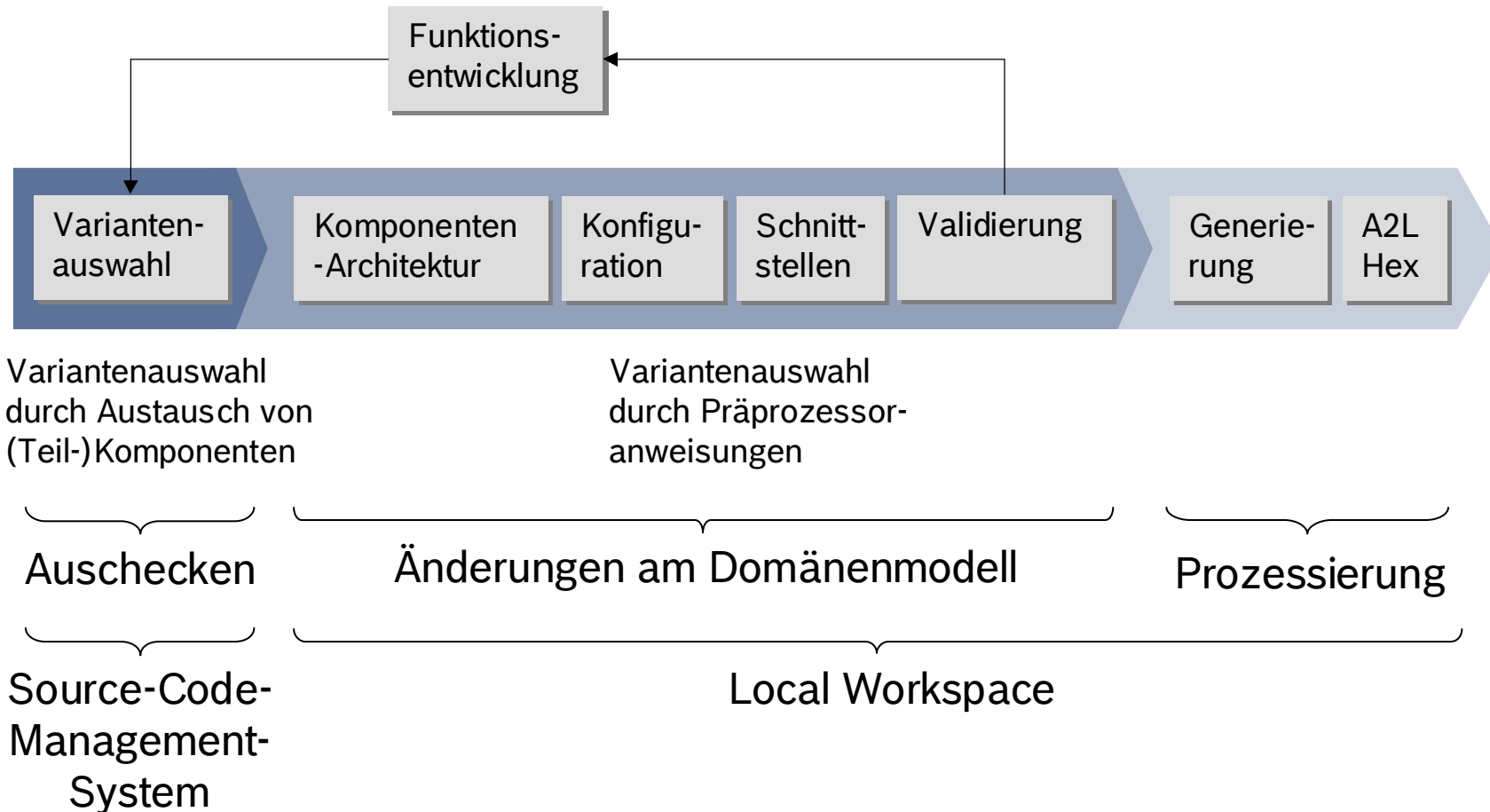
- Durchgängige, integrierte Werkzeugkette basierend auf einem gemeinsamen Domänenmodell
- **Effizienzsteigerung**
  - Verbessert „User Experience“
    - Eine einzige Umgebung mit einheitlicher Bedienung
    - Auswirkungen von Modelländerungen sofort sichtbar
  - Globale Optimierungsstrategien für Hardware-Ressourcen
  - Weniger Entwicklungs- und Wartungsaufwand für die Werkzeugkette
    - Gemeinsames Domänenmetamodell für alle Werkzeuge
- **Qualitätssteigerung**
  - Frühzeitige Validierung des Domänenmodells in allen Projektphasen
  - Sicherstellung der Systemintegrität während der gesamten Entwicklung
- **Kosteneinsparung**

## Agenda

- Motivation
- Der Steuergeräte-Software-Entwicklungsprozess
- Technische Grundlagen
- Unterstützung des Entwicklers
- Fragen und Diskussion

# Modellgetriebene Steuergeräte-Entwicklungsumgebung

## Steuergeräte-Software-Integrationsprozess



## Agenda

- Motivation
- Der Steuergeräte-Software-Entwicklungsprozess
- Technische Grundlagen
- Unterstützung des Entwicklers
- Fragen und Diskussion

## Technische Basis

- Basierend auf Eclipse 3.4
- Domänenmodell basiert auf dem Eclipse Modeling Framework (EMF)
- Validierung basiert auf dem EMF Validation Framework
- C-Funktionsentwicklung mit Eclipse C Development Tooling (CDT)
- Generierung mit openArchitectureWare (oAW)
  
- Eigenentwicklung
  - zur Integration der Werkzeuge
  - für domänenspezifische Anpassungen

## Source-Code-Management

- Domänenspezifisch konfigurierbares SCM
- Erweiterbarkeit bzgl. Domäne und Prozessen
  - Versionskontrolle von Gruppen und Elementen
  - Typisierung durch Klassifizierung
    - Klassen bilden Struktur der Software ab (z.B: Abliefereinheit, Architektur) und erlauben Grundvalidierung
  - Erweiterte Funktionalität durch Plug-Ins
    - Z.B. zusätzliche domänen- oder prozessspez. Validierungen
- Unterstützt Produktlinien mit hoher Varianz:
  - Einfacher Varianten/Branch-Mechanismus
  - Gruppen in Gruppen-Konzept
  - Gleicher Variantenmechanismus für Selektion im SCM wie auch für Präprozessorvarianz genutzt

## Local Workspace

- Schnittstelle zwischen SCM-System und Entwicklungsumgebung/externen Werkzeugen
  
- Motivation
  - Bereitstellung der durch das SCM System bekannten Mechanismen lokal auf der Platte
    - Klassen
    - Metainformationen
  - Kommunikation zwischen lokalen Tools auf dieser Basis
  
- Einbettung in die IDE
- Bereitstellung einer API (Java und Kommandozeile)

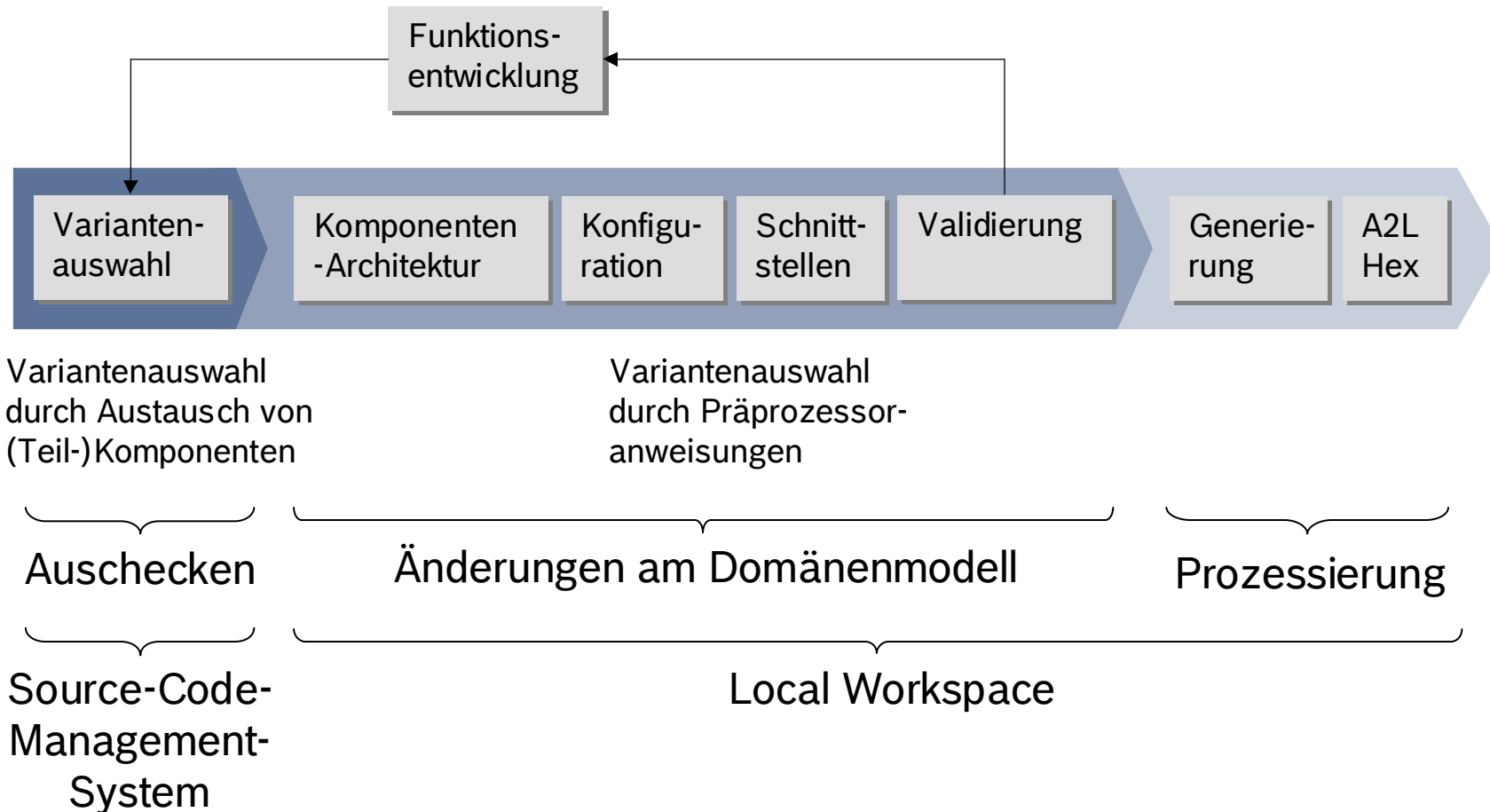
## Gemeinsames Domänenmodell

- Gemeinsames Domänenmodell für Software-Metadaten (Komponenten, Schnittstellen, Typen, Dokumentation...)
- Enthält **Produktlinie**
- Konkretes **Produkt** wird im Domänenmodell **berechnet**
- Verwendung des Domänenmodells
  - für formular-basierte Komponenten-Editoren
  - zur Validierung der Komponenten
  - zur Unterstützung der Funktionsentwicklung im C-Editor
  - zur Generierung der C-Header-Dateien
  - zur Konfiguration der Software
- Technische Besonderheiten:
  - Gute Skalierbarkeit, Modelle von mehreren 100MB üblich
  - Gute Navigierbarkeit, Namensref. aus Dateiartefakten aufgelöst

## Agenda

- Motivation
- Der Steuergeräte-Software-Entwicklungsprozess
- Technische Grundlagen
- Unterstützung des Entwicklers
- Fragen und Diskussion

## Steuergeräte-Software-Integrationsprozess



## Komponentenbearbeitung

- Allgemeine XML-Texteditoren für Metadaten
  - Keine Einschränkung der Eingaben
  - Für erfahrene Benutzer häufig schneller zu bedienen (Copy & Paste)
- Domänen-spezifische Formular-Editoren
  - Bereiten Daten zur besseren Übersicht auf
  - Beschränken Eingaben des Benutzers auf gültige Werte
  - Stellen vordefinierte Werte zur Verfügung
- Automatische Synchronisation zwischen XML- und Formular-Editoren
- Synchronisation mit dem C-Modell
- Navigatoren auf Komponenten-Basis
  - Abstraktion vom Dateisystem

## Produktkonfiguration

- Berechnung eines Produktes aus der Produktlinie
- Konfigurierbare Software-Komponente enthält
  - Templates (C-Sourcecode, Metadaten)
  - Konfigurationsregeln
  - Generator
- **Konfigurationsregeln** (Metadaten)
  - Beschreiben Konfigurationsparameter für eine Komponente
  - Beschreiben Abhängigkeiten der Parameter untereinander
  - Enthalten Informationen für Konfigurationseditoren
  - Definiert durch den Entwickler der Komponente
- **Konfigurationsdaten**
  - Entsprechen den Konfigurationsregeln
  - Festgelegt durch den „Benutzer“ der Komponente
- Generator erzeugt aus Templates und Konfigurationsdaten die konkrete Komponente

## Modellvalidierung

- Frühzeitige Validierung bereits während der Entwicklung
- Gesamtes Modell wird validiert
- **Inkrementelle** Modellvalidierung angestoßen durch Modelländerungen
- **Vollständige** Modellvalidierung angestoßen durch den Benutzer
- Validierungsprobleme sichtbar in Editoren und Problems View
- Reporting der Ergebnisse in PDF, HTML, Excel, Word
- Mehr als 350 Regeln aus den verschiedensten Bereichen
- Modellübergreifende Validierung möglich
  - C vs. Metadaten
  - Dokumentation vs. Metadaten
- Kommandozeilen-basierte Version für automatisierten Softwarebuild

## C-Funktionsentwicklung

- Grundlage: Eclipse CDT Umgebung
- Implementierung gegen Domänenmodell, nicht gegen C-Header
- Indexierung über C-Modell hinaus auf Domänenmodell ausgedehnt
  - Code-Vervollständigung mit Hilfe des Domänenmodells
  - Suche nach Deklarationen von Funktionen und Variablen im Domänenmodell
  - Verlinkung von Funktionen und Variablen im C-Editor mit ihren Deklarationen
- Modellübergreifende Validierung von C-Modell der Funktionen und des Domänenmodells

## Generierung

- Basis ist das Domänenmodell
- C-Header für Funktionskomponenten
- Core-Software (gemeinsame Basisfunktionen für unterschiedliche Steuergeräte)
  - Basis sind zusätzlich C-Grundgerüste und Konfigurationsdaten
  - Vollständige, kompilierbare Software
- Synchronisationsmechanismen für globale Variablen
  - Schnittstellen zwischen Komponenten sind globale Variablen
  - Variablen werden bei konkurrierenden Zugriffen kopiert
- A2L-Daten für Kalibrierungsparameter aus Kompilat und Domänenmodell
  - Position der Kalibrierungsdaten im Kompilat
- Metadaten zur Fehlerdiagnose/zum Auslesen des Fehlerspeichers eines Steuergerätes

### Software-Build (in Entwicklung)

- Integriert in die Entwicklungsumgebung und Kommandozeilen-Version
- Verschiedene Datenquellen
  - Dateien
  - Domänenmodell
  - Datenbanken
- Kontrollfluss wird zum Teil aus Daten abgeleitet
- Inkrementeller Build bei Änderungen an den Daten
- Partieller und vollständiger Build auf Anforderung
- Parallelisierung von Prozessschritten

## Projekt-Dokumentation

- Gesamte Dokumentation wird im Domänenmodell hinterlegt
- Dokumentenkategorien dienen zur Strukturierung
  - Entwicklerdokumentation
  - Kalibrierungsdokumentation (für Anwender der Komponente)
  - Komponenten-Übersicht
  - Beliebig erweiterbar durch neue Kategorien
- Varianten werden berücksichtigt und aufgelöst
- Mehrsprachenfähig
- Generierung der Dokumentation
  - einer einzelnen Komponente
  - einer Komponente inklusive ihrer kompletten Substruktur
  - des vollständigen Projekts
  - nach Zielgruppe (Entwickler, Anwender, Architekten,...)

## Agenda

- Motivation
- Der Steuergeräte-Software-Entwicklungsprozess
- Technische Grundlagen
- Unterstützung des Entwicklers
- Zusammenfassung und Ausblick

## Zusammenfassung und Ausblick

- Gemeinsame IDE-Basis
  - Erstes Rollout Q4/2009
  - Weitere Funktionsgruppen werden iterativ integriert
  - Geschäftsbereich-spezifische Erweiterungen
- 
- Große Herausforderung: AUTOSAR