Embedded Systems
Konzepte moderner Software-Ecosysteme

Lehrstuhl für Informatik 4
Hauptseminar AKSS
10.01.2013

Denis Koslowski
koslowski.d@web.de

Friedrich-Alexander-Universität
Erlangen-Nürnberg

Inhaltsverzeichnis

Motivation

Embedded System

Definition: *„...heterogene technische Systeme, die sich durch verschiedenartige Komponenten und Interaktionen auszeichnen, die auf einen ganz bestimmten Anwendungsbereich zugeschnitten sind und die in einem technischen Kontext eingebettet sind...“* [1]

Aufgaben: Überwachungs- Steuerungs- und Regelfunktionen, Daten- oder Signalverarbeitung [2]

Verbreitung: „...mehr als 99% aller Prozessoren sind in eingebetteten Systemen verbaut...“ [3]

Motivation

Fahrzeuge

Haushaltsgeräte

Medizintechnik

Telekommunikation

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

Motivation

Charakteristische Merkmale von Embedded Systemen:

- tiefe Integration zw. Software und Hardware
- starker Fokus auf Herstellungsprozess
- große Beteiligung der Zulieferer
- sicherheitskritische Funktionen

Motivation

Probleme bei der Entwicklung:

- Zeitdruck
- Komplexität der Anforderungen
- Kosten
- fehlende Kompetenz

Inhaltsverzeichnis

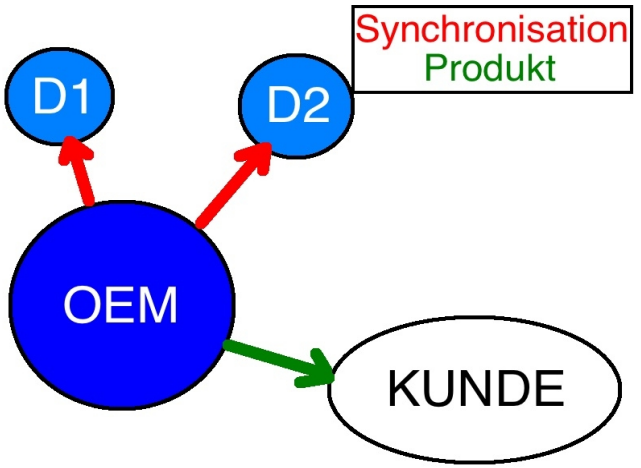
Aktuelle Entwicklungsmethode

Intergrationszentrische Methode

Bei komplexen Embedded Systemen (z.B. Autos) ist es üblich ein Teil von der Softwareentwicklung auszulagern.

OEM synchronisiert die Entwicklung zwischen allen beteiligten Parteien.

OEM hat volle Kontrolle über SW-Inhalt des Endproduktes.



Aktuelle Entwicklungsmethode

Intergrationszentrische Methode

Nachteile der Methode bei starker Auslagerung:

- Kontrolle von externen Entwicklern ist kaum möglich
- Immer kürzer werdende Markteinführungszeit
- Mehr Funktionen (SW-Teile) aber kurze Integration
- schlechte Kommunikation zwischen den Entwicklerteams

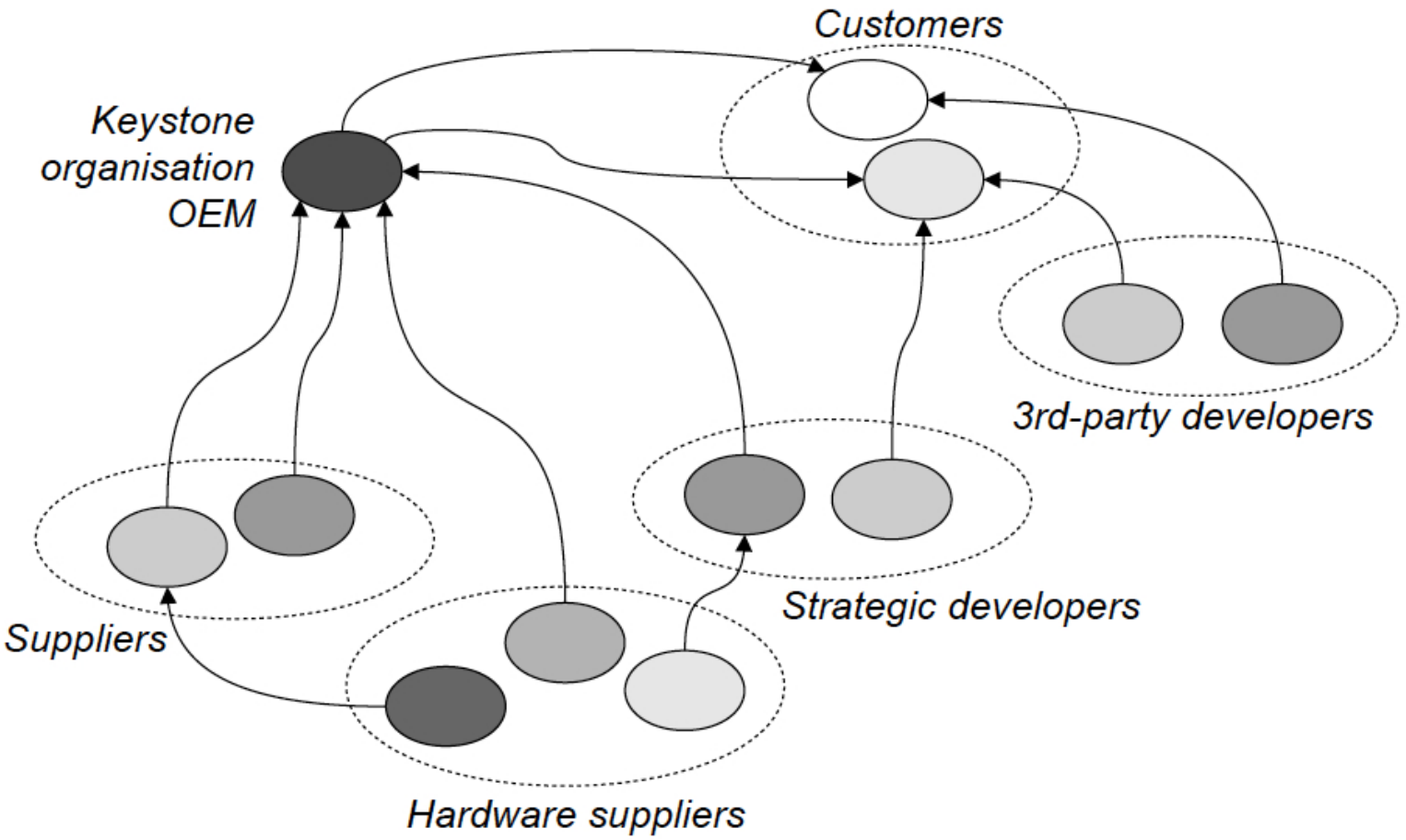
Aktuelle Entwicklungsmethode

Frage: Gibt es eine bessere Methode für die Entwicklung von Embedded Software?

Antwort: JA. Software Ecosystem.

Inhaltsverzeichnis

Ecosystem für Embedded Software



[4]

Stakeholders

OEM *(Original Equipment Manufacturer)*

- Keystone-Player (hat entscheidende Rolle)
- Entwickler der Ecosystem-Plattform
- verantwortlich für die kritische Applikationen
- zuständig für die Stabilität und Weiterentwicklung des Ecosystems

Strategische Entwickler

- haben Langzeit-Beziehungen oder Partnerschaft mit OEM
- erweitern strategische Bereiche des ES (gelenkt von OEM)
- haben einen tiefen Zugriff zur ES-Plattform
- liefern zuverlässige Software (nach Zertifizierung)

Stakeholders

Drittentwickler

- keine direkte Geschäftsbeziehung mit OEM
- Geschäftsausrichtung auf Endkunden
- eingeschränkte Zugriff auf kritische Funktionen des System
- bringen signifikante Erweiterung der Funktionen

Zulieferer

- werden durch OEM beauftragt
- entwickeln neue oder oder pflegen alte Applikationen

Qualitätsattribute eines Ecosystems

Notwendige Qualitätsattribute:

- Zusammensetzbarkeit
- Einsatzfähigkeit
- Wartbarkeit
- Konfigurierbarkeit

nicht universelle aber nützliche in manchen Domänen:

- Konsistentes User-Interface
- Zuverlässigkeit

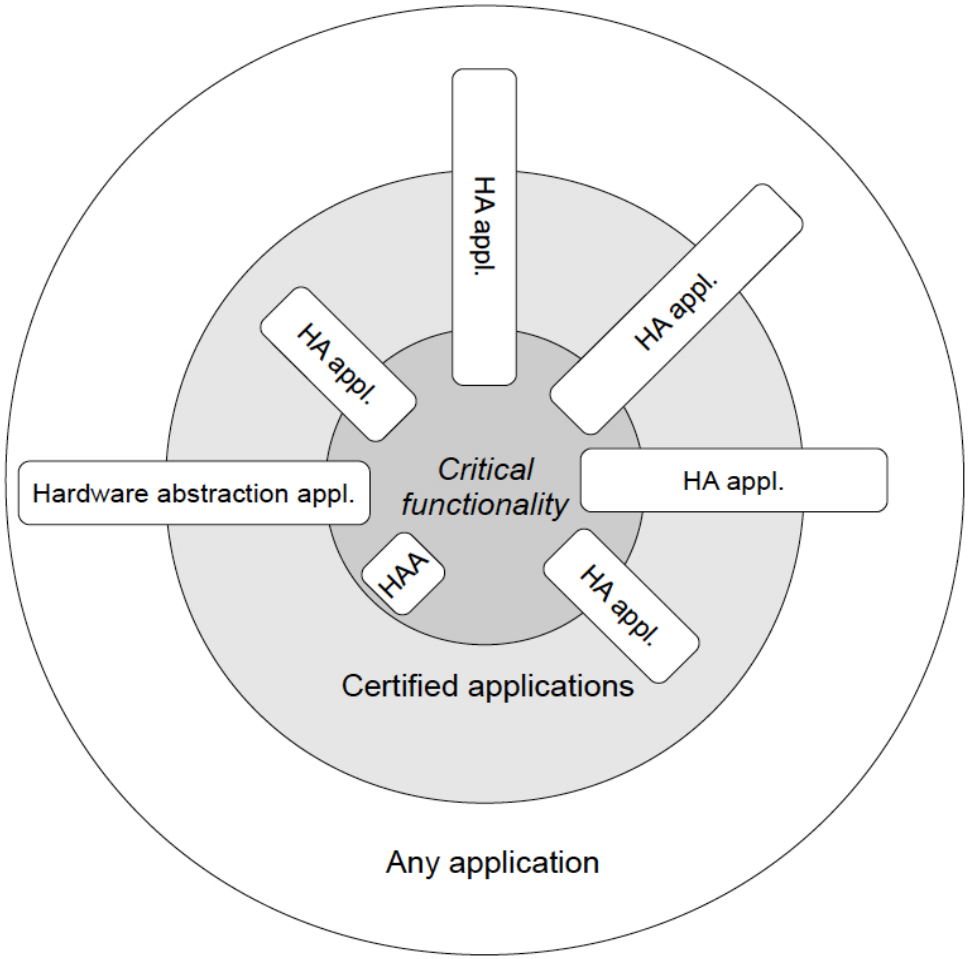
Multiple Levels of Access

Nicht alle Applikationen sind zuverlässig.

Zugriff auf die Sensoren und Aktoren eines Embedded Systems bringt Gefahren und Risiken und kann zum System-Missbrauch oder zum System-Versagen führen.

Zugriff auf die kritische Hardware soll durch die spezielle Mechanismen geregelt werden

Multiple Levels of Access



[5]

- **Kritische Funktionalität**
  - Zugriff auf kritische Aktoren und Sensoren (hardware abstraction applications - HAA)
  - isolierten Umgebung
- **Zertifizierte Applikationen**

  erfüllen Sicherheitsprozesse und Zuverlässigkeitsanforderungen
- **Beliebige Applikationen**

FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG

Ecosystem für Embedded Software

Ecosystem bringt folgende Vorteile für OEM:

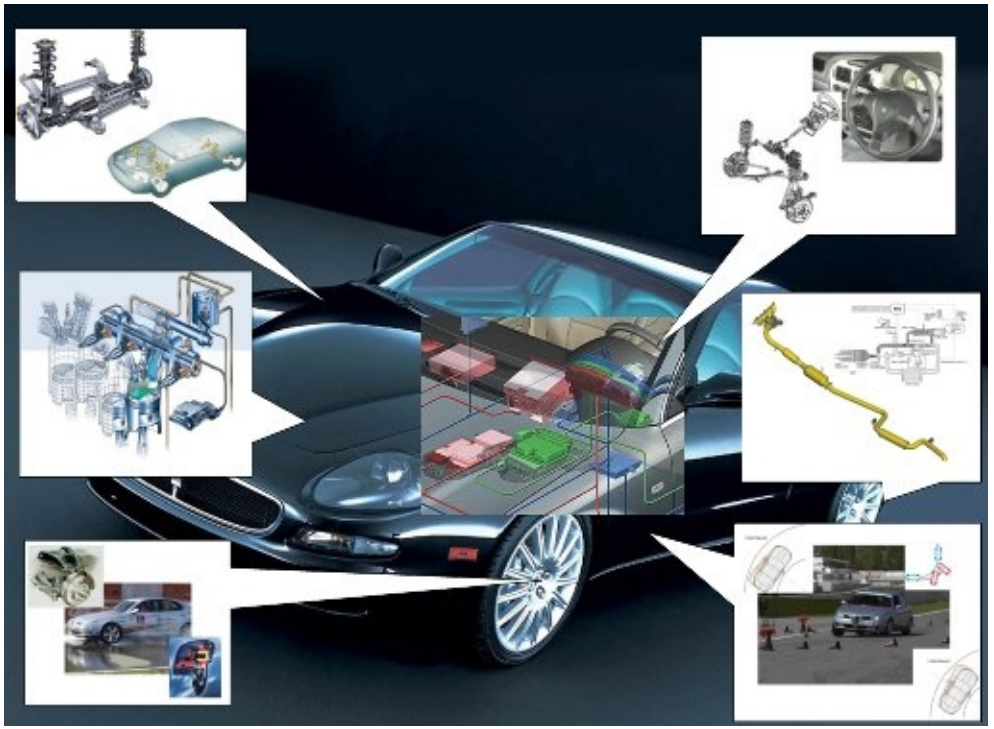
- Synchronisierung von Entwickler durch die Architektur
- Unterstützung von Zuverlässigkeitsanforderungen
- Schutz der kritischen Funktionalitäten
- Entkoppelung zwischen Applikationen → Funktionenzuwachs
- Entkoppelung in der Zeit → jederzeitige Softwareerweiterung des Produkts

Inhaltsverzeichnis

Beispiel: Autoindustrie

- Auto: das komplexeste Produkt mit Embedded Software
- 50-100 Entwicklerteams nur für Embedded Software
- Große Anzahl von Steuergeräten (ECU – electronic control unit) für die Steuerung von Sensoren und Aktoren

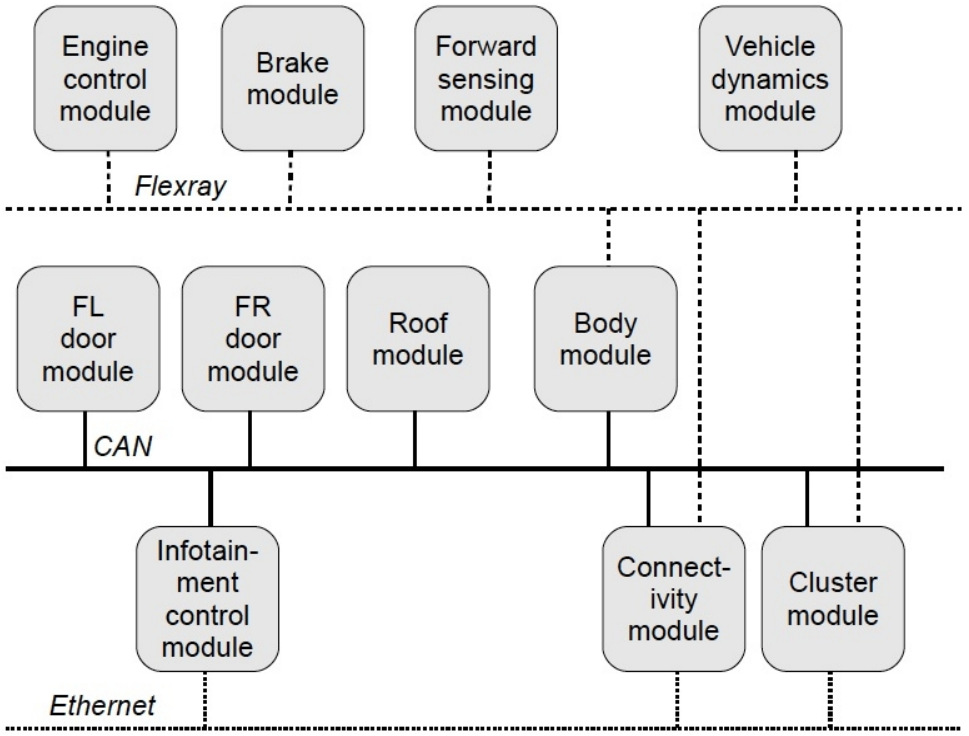
Beispiel: Autoindustrie



Ecosystem für Autoindustrie:

- OEM liefert/produziert Autos mit minimaler SW-Konfiguration (Motor-, Chassis-, Display-Steuerung)
- Vertriebspartner können zusätzliche SW-Funktionen einbauen (Infotainment, Body&Comfort-Funktionen)
- Drittentwickler können ihre Applikationen direkt an Endkunden liefern.

Beispiel: Autoindustrie

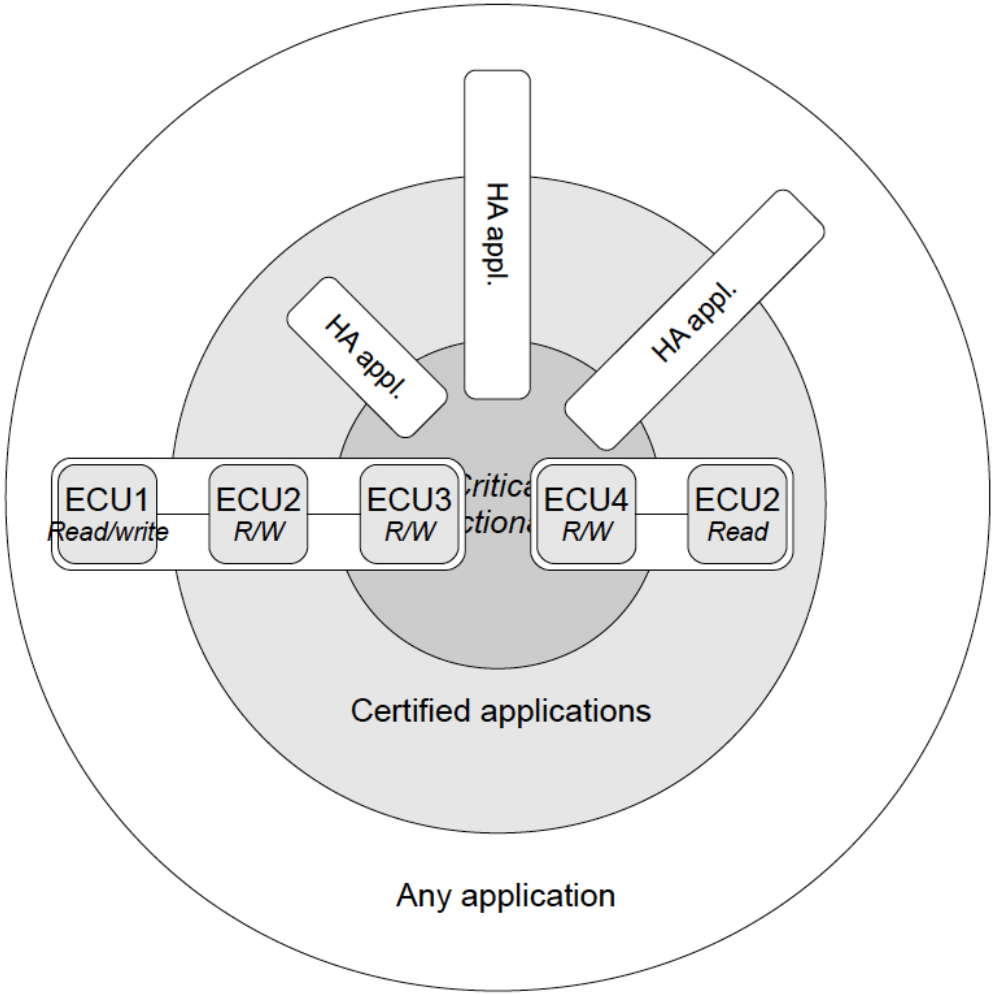


[6]

- HMI, Telematics und Multimedia-Funktionen können komplett offen sein (keine Zertifizierung nötig)
- Antrieb- Fahrwerk- Sicherheit-Funktionen sind kritisch und müssen zertifiziert werden
- Body&Comfort Sensoren und Aktoren - Zertifizierung beim schreibenden Zugriff notwendig
- Connectivity Modul kontrolliert Zugriffe auf kritische Module, überprüft Zertifizierung von Applikationen

FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG

Beispiel: Autoindustrie



[7]

- Kritische Funktionen werden durch die separate ECU's ausgeführt
- ECU's kommunizieren durch Daten-Busse (Flexray, CAN, Ethernet)

Kurze Zusammenfassung

- Ecosysteme sind geeignet für die Entwicklung von software-intensiven Embedded Systemen
- Besser als integrationszentrische Methode
  - Synchronisationsbedarf ist niedrig
  - breiteres Spektrum von Funktionen möglich
  - Schnellere Entwicklung
  - Sicherheit und Zuverlässigkeit des Systems kann durch Architektur der Ecosystem-Plattform vorgegeben werden
- Auto ist ein sehr komplexes software-intensives Embedded System → Ecosysteme werden zukünftig eine große Rolle für die Autoindustrie spielen

Literaturhinweise

- [1], [3] Jürgen Teich, Christian Haubelt, „Digitale Hardware/Software Systeme, Synthese und Optimierung“, 2007, p. 1
- [2] http://de.wikipedia.org/wiki/Eingebettetes_System
- [4], [5], [6], [7] U. Eklund and J. Bosch, "Using architecture for multiple levels of access to an ecosystem platform," Proceedings of the 8th international ACM SIGSOFT conference on Quality of Software Architectures - QoSA '12, p. 143, 2012.

FRAGEN?

Vielen Dank für Ihre Aufmerksamkeit!