

Process-oriented System Development PSD - Konstruktive Entwicklungsmethode für Embedded Systeme

PSD ist eine Methode, mit der Embedded Systeme komponentenbasiert und architekturbezogen entwickelt werden. Die Methode umfasst sowohl die modellbasierte Konstruktion der Steuerfunktionen (CIP) als auch das Scheduling und das I/O-Handling für die aus den Modellen generierten Komponenten.

PSD ist für den Bau zuverlässiger Systeme ausgelegt. Die Entwicklungskonzepte helfen dem Anwender der Methode, die entstehende Systemkomplexität unter Kontrolle zu halten. Es werden keine versteckten Abhängigkeiten zwischen Komponenten zugelassen, und die Ausführung eines Systems erfolgt so deterministisch wie möglich (nur wenige asynchrone RTOS-Tasks).

Konstruktionstechnik: Architekturorientierte Komposition

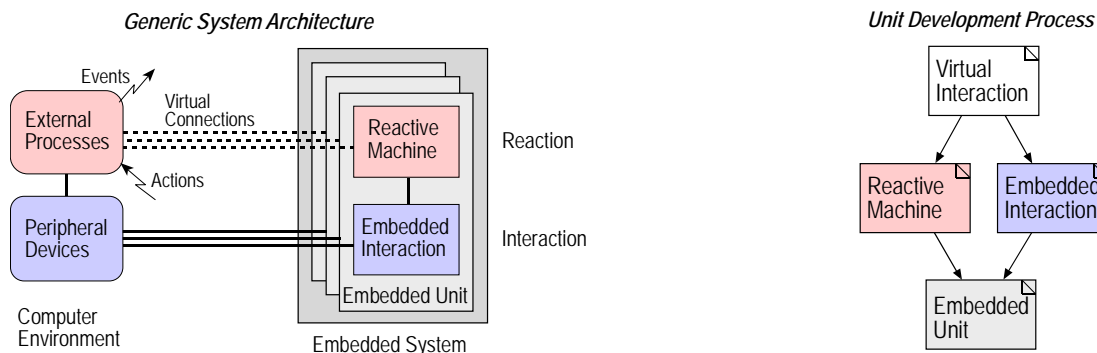
Die Komposition von *Komponenten* erfolgt mit PSD immer über *Konnektoren*, die in entsprechenden Architekturmodellen definiert sind. In Komponenten werden Funktionen und Abläufe definiert, und mittels Konnektoren wird die Interaktion zwischen den Komponenten spezifiziert. Konnektoren sind entweder elementare Bauteile, die durch ein entsprechendes Framework zur Verfügung gestellt werden (fine grained modelled composition), oder sie werden vom Entwickler spezifiziert und als komponenten-basiertes Subsystem implementiert (coarse grained composition).

Mit dieser konstruktiven Komponententechnik lassen sich robuste Software-Systeme mit flexibler Funktionalität bauen. Wie die Erfahrung zeigt, lässt sich z. B. der Wartungs- und Weiterentwicklungsaufwand um bis zu 50% reduzieren.

Separation of Concerns - Virtuelle Interaktion als prozessorientiertes Entwicklungskonzept

“Components are Abstractions with Plugs” (O. Nierstrasz)

Um die “richtigen” Abstraktionen für Embedded Systeme zu finden arbeitet man in PSD mit dem prozessorientierten Konzept der *virtuellen Interaktion*. Das Konzept ermöglicht funktionale Lösungen (*Reactive Machine*) und deren Einbettung auf den Zielrechnern (*Embedded Interaction*) streng getrennt zu entwickeln.



Prozessorientierte Topstrukturen: generische System-Architektur und Entwicklungsprozess einer Unit

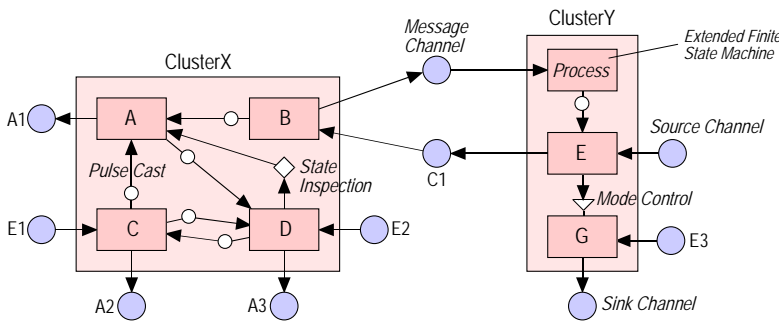
Die virtuelle Interaktionsverbindung zwischen *Externen Prozessen* und *Reaktiven Maschinen* wird für die zu verarbeitenden Prozessereignisse (*Events*) und die bei den Prozessen zu bewirkenden Aktionen (*Actions*) spezifiziert. Sie ermöglicht, die Steuerfunktionen unabhängig von den *Peripheriegeräten* als reaktive Modelle zu entwickeln. Implementiert wird die virtuelle Verbindung durch die installierten *Peripheriegeräte* und die parallel zum funktionalen Modell entwickelte *Embedded Interaction Software*.

Die Beschreibung der *virtuellen Interaktion* auf der Basis von *Events* und *Actions* erweist sich für den ganzen Entwicklungsprozess als Schlüsselspezifikation:

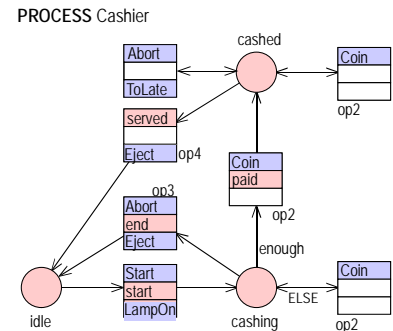
- sie definiert die Abstraktionsebene für die funktionale Entwicklung
- sie spezifiziert die Einbettung der generierten Komponenten
- sie ermöglicht parallele Threads im Entwicklungsprozess
- sie entflechtet funktionale und zeitliche Anforderungen

Reaktive Maschine - *CIP-Modell (Funktionale Anforderungen)

*CIP Modelling Framework unterstützt durch CIP Tool® <http://www.ciptool.ch>



Architektur eines CIP-Modells



Modus einer Zustandsmaschine

Die CIP-Methode wurde schon früher für die prozess-orientierte Konstruktion von Steuermodellen konzipiert und ist jetzt entsprechend in PSD integriert.

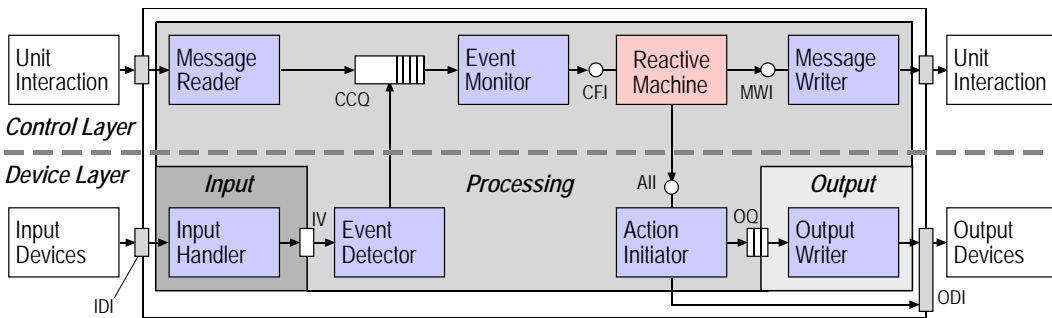
CIP Modelling Framework

- architekturorientierte Komposition von Zustandsmaschinen (Konnektoren als Interaktionsmodell)
- Asynchrone Komposition von Clustern
- Cluster sind synchrone Kompositionen erweiterter Zustandsmaschinen
- Generierung nebenläufig ausführbarer reaktiver Komponenten (*Reaktive Maschinen*)

CIP Modellierungsprozess

- Schritt 1 - Prozessagenten garantieren Verhaltenskorrespondenz mit den externen Prozessen
- Schritt 2 - Grundlegende Steuerfunktionen durch Komposition der Agenten mit Funktionsprozessen
- Schritt 3 - Erweiterte Funktionen durch bilden von Mode-Control-Hierarchien

Embedded Interaktion (Zeitliche Anforderungen)



Generische Architektur einer Embedded Unit

Die aus den Modellen generierten Software-Komponenten (*Reaktive Maschinen*) werden real mit den externen Prozessen verbunden, indem die virtuelle Interaktionsverbindung mittels Sensoren und Aktoren und entsprechender Softwareinteraktion (*Embedded Interaktion*) realisiert werden.

Generische Komponenten der Embedded Interaktion einer Embedded Unit:

- Zugriffe auf Eingabe- und Ausgabe-Schnittstellen (Sensoren, Aktoren, Busse)
Input Handler und *Output Writer* (hardware-nahe Funktionen)
- Interpretation der Input-Daten und bereitstellen der Output-Daten
Event Detector und *Action Initiator*
- Ablaufkontrolle
Ein *Event-Monitor* steuert zentral die non-preemptive Ereignisverarbeitung (Kombination von *FIFO* und *Deadline Scheduling*)

Scheduling mehrere Embedded Units (asynchrone RTOS-Tasks)

- Embedded Units* werden als periodische Server für sporadische Ereignisse ausgeführt (Rate Monotonic Scheduling Policy with Deferred Capacity Intervals)