Modellbasiert zum Seriencode

Zwei Projekte – zwei Tools – viele Erkenntnisse

Joachim Terasa • coming GmbH

Modellbasierte Softwareentwicklung?

- ständig wachsende Komplexität
- erhöhte Anforderungen an Dokumentation
- Modell als Kommunikationsmittel

Modellbasierte Softwareentwicklung!

- mit Codegenerierung

MBSE ist für embedded Systeme Realität

Konventionelle Entwicklung vs. MBSE

Toolchain konventionell

- Sourcecode- und Versionsverwaltung
- Editor
- Compiler/Linker/Loader
- Betriebssystem
- Debughilfsmittel
- Testhilfsmittel

Toolchain MBSE/Codegenerator

- Sourcecode- und Versionsverwaltung
- Modellierungswerkzeug
- Compiler/Linker/Loader
- Betriebssystem
- Debughilfsmittel
- Testhilfsmittel

Vom Editor zum MBSE-Tool

Editor

- Erstellen von Sourcecode
- Formatierfunktionen
- Suchfunktionen

100% Programmierer

MBSE-Tool

- Erstellen eines Designs
- Textuelle und grafische Bestandteile
- Codegenerierung
- Verbindung zu Test und Debugging

XX% Codegenerator
YY% Programmierer

Auswahlverfahren

- **Evaluierung**
  - **Bedienbarkeit, Benutzerfreundlichkeit**
  - **Funktionsumfang**
  - **Musterprojekt**
  - **Codeanalyse**
  - **Codeumfang**
  - **Performancemessungen**

- **Feature Vergleich**

	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	F16	F17	F18	F19	F20	F21	F22	F23	Sum
Wichtung	3	2	1	1	1	4	3	1	1	1	1	2	1	5	5	2	2	1	1	1	2	3	1	1
Tool1	1	3	1	2	2							2	3	2	1	1	2	2	3			1	1	51
Tool2					2		1	2	1	3	1	1	3	3	3	3	2	2	1	3	2	3		76
Tool3		3				1	2		2	1	1		2	1	2		2	2	2	1	3	2	1	59

Zwei Projekte – zwei Tools – viele Erkenntnisse

radCASE

Hersteller: IMACS

www.imacs-gmbh.de

Payware

Hersteller: Quantum® LeaPs innovating embedded systems

www.state-machine.com

Freeware/Payware

Gemeinsamkeiten

System

- für embedded Systeme
- Generierung von C-Code
- direkt targetfähig (Toolchain)
- BSP bzw. API muss erstellt werden

Gemeinsamkeiten

Modell

- **UML basiert**
- **hierarchische Statemachines**

Gemeinsamkeiten

Architektur

- **Objektorientiert**
- **keine Klassendiagramme**
- **Objektbaum**

Gemeinsamkeiten

Ausführung

- **Tasking**
- **kooperativ**
- **RTC (Run to completion)**

Gemeinsamkeiten

Entwicklerunterstützung

- **Simulation am PC**
- **Debugunterstützung am PC**

Gemeinsamkeiten

Projektorganisation

- **Partitionierung von Projekten**
- **Projekte in XML-Dateien**
- **Proprietäre Editoren**

„Gemeinsamkeiten“ im Detail

Wie spricht die Software mit der Hardware?

radCASE

- **API mit definierter Schnittstelle**
  - **HW mit API austauschbar**
  - **Modell bleibt unverändert**
  - **Windows-API**

- **individuelles BSP**
  - **Trennung BSP/Modell willkürlich**
  - **Geänderte HW ► neue SW**

„Gemeinsamkeiten“ im Detail

Wie spricht die Software mit der Hardware?

radCASE

- **SW-HW-Mapping mit „Entity-Tab“**
- **Bit-Zugriffe durch Codegenerator**

Definition:

```
BL_red
BL_green
BL_blue
```

Code:

```
$BL_red = §OFF;
$BL_green = §ON;
$BL_blue = §OFF;
```

Entity-Tab:

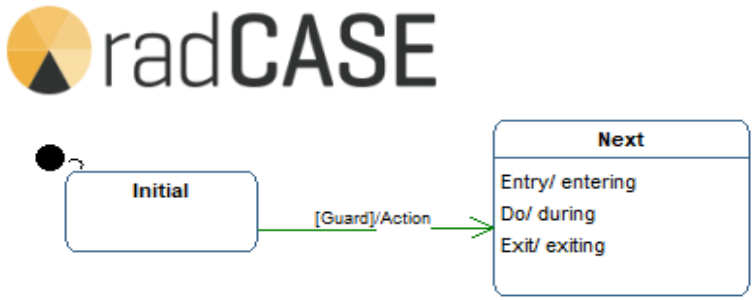
```
BL_red: CTR RG,0,4,NINV,0,0
BL_green: CTR RG,0,5,NINV,0,0
BL_blue: CTR RG,0,6,NINV,0,0
```

- **Standard C-Code**
- **Bit-Zugriffe über Masken**
- **HW-Zugriff über Instanz-ID**

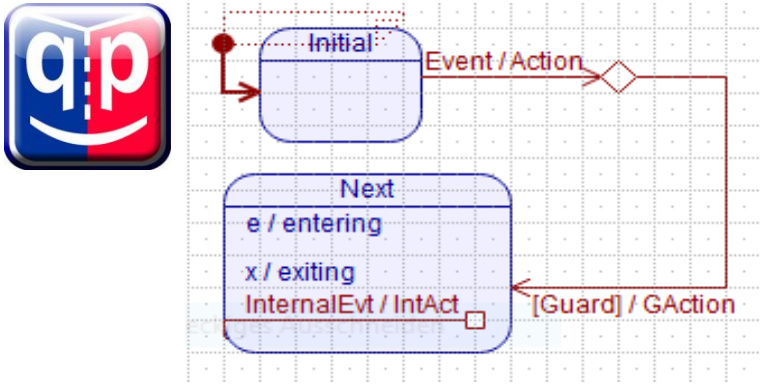
```
if (instance == 0) switch_pump1(OFF); // first instance
else               switch_pump2(OFF); // second instance
```

„Gemeinsamkeiten“ im Detail

Wie funktionieren Statemachines?



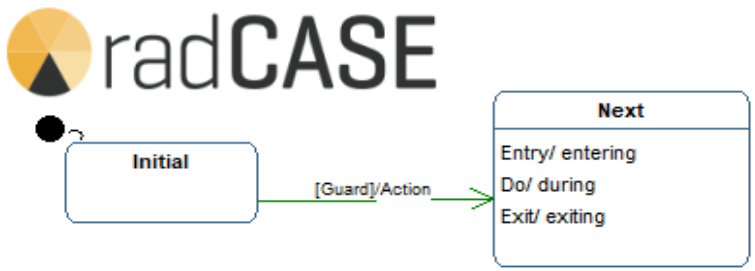
- **polled statemachine: getaktet**
- **zyklische Ausführung**
- **Actions = C-Codefragmente {}**
- **Do-Action solange Next aktiv**
- **Keine concurrent states**



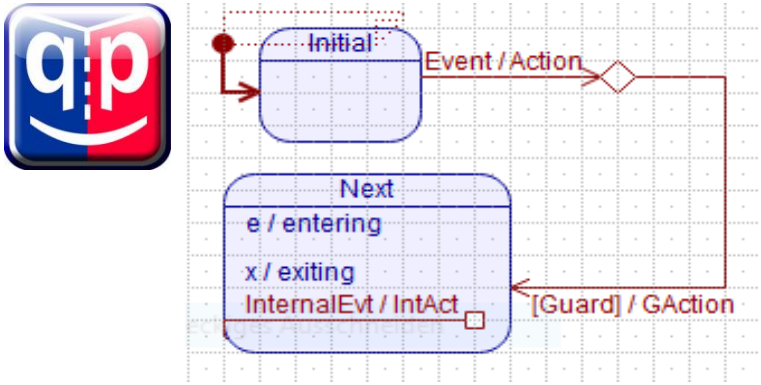
- **event-driven statemachine: asynchron**
- **ohne Event keine Ausführung**
- **Actions = C-Codefragmente {}**
- **IntAct bei jedem InternalEvt in Next**

„Gemeinsamkeiten“ im Detail

Wie funktionieren Statemachines?



- **zyklisch, nicht garantierte Reihenfolge**
- **[Guard] nicht zwangsläufig konsistent**



- **Prioritätsgesteuerte Reihenfolge**
- **Events enthalten Daten**
- **Event Queues**

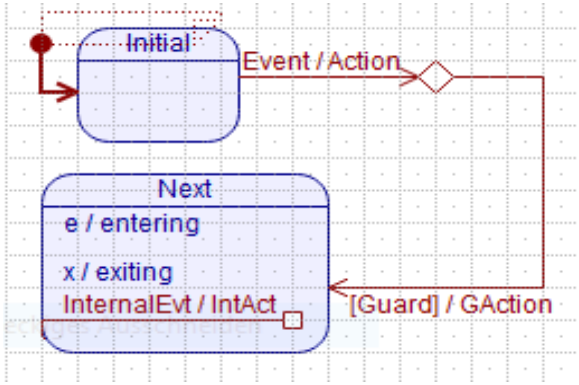
„Gemeinsamkeiten“ im Detail

Wie funktioniert Tasking?

radCASE

Function name:	DisplayColorAndLEDs
Interface type:	
Processing type:	PERM
Return type:	void

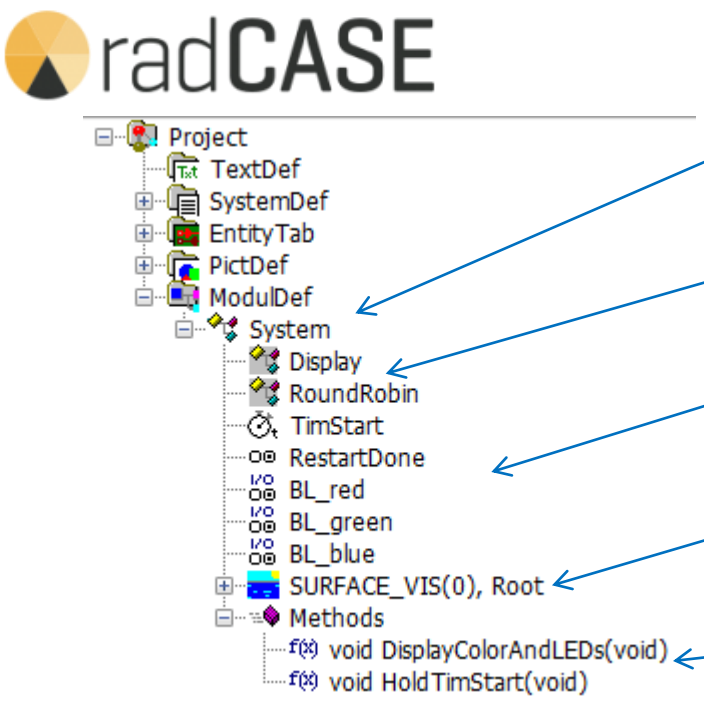
- **zyklisch**
- **jede „PERM“ Funktion und Statemachine**
- **unterschiedliche Zykluszeiten möglich**
- **kooperativ**



- **kein Tasking im herkömmlichen Sinne**
- **vollständig statemachine-basiert**
- **vollständig asynchron**
- **Actions sind RTC**

„Gemeinsamkeiten“ im Detail

Objektorientierung



Definition des Objektes „System“

Subobjekte (Instanz)

Attribute

GUI

Methoden

„Gemeinsamkeiten“ im Detail

Objektorientierung

External Package „Inputs“

Active Object AO_Inputs

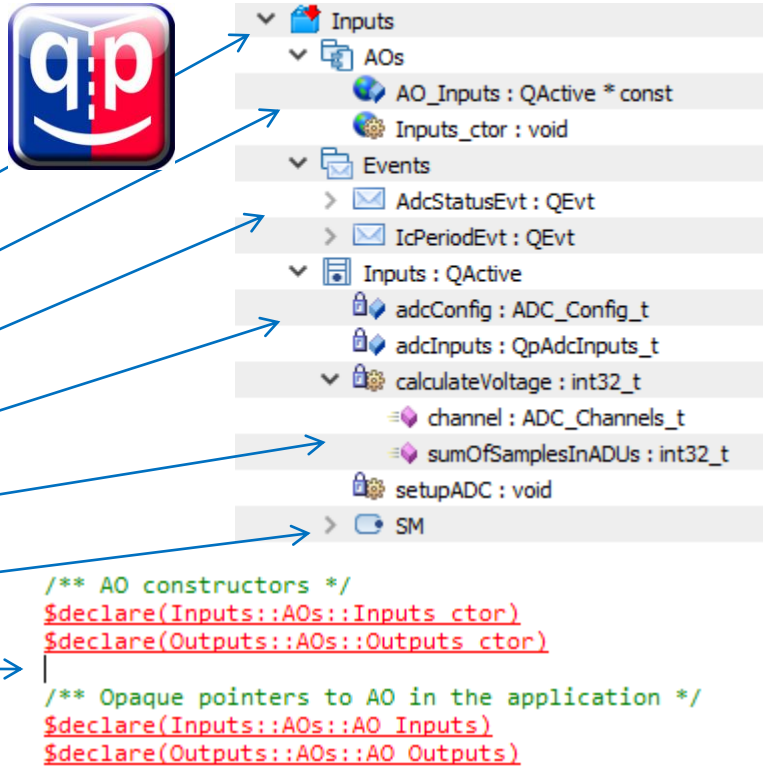
Events

Attribute

Methoden

Statemachine

Instanziierung



„Gemeinsamkeiten“ im Detail

Simulation und GUI

- **QWIN nur zur Simulation**
- **BSP für Simulation muss erstellt werden**
- **30-seitige Step-by-step-Anleitung**
- **Visual C/C++**

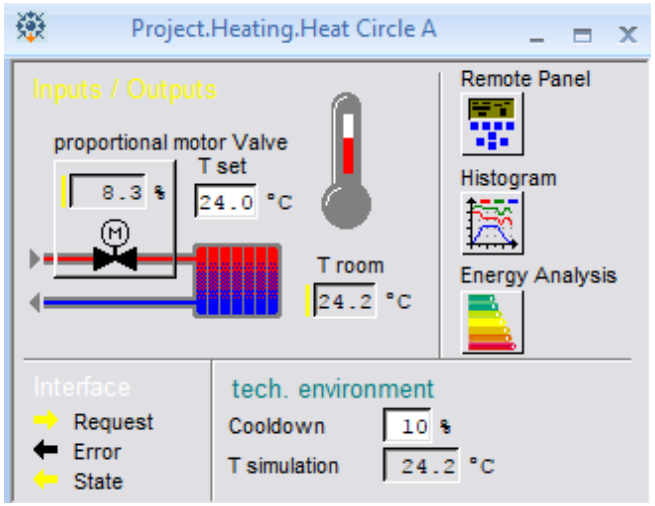
„Gemeinsamkeiten“ im Detail

Simulation und GUI

radCASE

- Eigene Beschreibungssprache für GUI
- GUI für Simulation und Visualisierung
- GUI-Elemente sind Teil der Objekte
- Visualisierung: Target-Kommunikation
- Standard-GUI

- Ein Klick zur Generierung



„Gemeinsamkeiten“ im Detail

Debugging

```
           INIT : Obj=l_Mbs Source=QpMbs_txing Target=QpMbs_txRun
3100201988 User003: TX:   7  1  3  2  0  0 184 68
           ENTRY: Obj=l_Mbs State=QpMbs_txRun
3100201998 ==>Tran: Obj=l_Mbs Sig=MBS_FRAME_OK_SIG Source=QpMbs_rcving New=QpMbs_txRun
3100202158 User003: event = MBS_SIO_TX_FINISHED
           EXIT : Obj=l_Mbs State=QpMbs_txing
3100202184 ==>Tran: Obj=l_Mbs Sig=MBS_SIO_TX_FIN_SIG Source=QpMbs_txRun New=QpMbs_rcving
3102001735 User003: event = MBS_FRAME_OK
3102001741 User003: event = MBS_FRAME_OK and Good Request
           INIT : Obj=l_Mbs Source=QpMbs_txing Target=QpMbs_txRun
           ENTRY: Obj=l_Mbs State=QpMbs_txRun
3102001769 ==>Tran: Obj=l_Mbs Sig=MBS_FRAME_OK_SIG Source=QpMbs_rcving New=QpMbs_txRun
3102002135 User003: event = MBS_SIO_TX_FINISHED
           EXIT : Obj=l_Mbs State=QpMbs_txing
3102002161 ==>Tran: Obj=l_Mbs Sig=MBS_SIO_TX_FIN_SIG Source=QpMbs_txRun New=QpMbs_rcving
3103801915 User003: event = MBS_FRAME_OK and Good Request
           INIT : Obj=l_Mbs Source=QpMbs_txing Target=QpMbs_txRun
           ENTRY: Obj=l_Mbs State=QpMbs_txRun
3103801942 ==>Tran: Obj=l_Mbs Sig=MBS_FRAME_OK_SIG Source=QpMbs_rcving New=QpMbs_txRun
3103802309 User003: event = MBS_SIO_TX_FINISHED
           EXIT : Obj=l_Mbs State=QpMbs_txing
3103802335 ==>Tran: Obj=l_Mbs Sig=MBS_SIO_TX_FIN_SIG Source=QpMbs_txRun New=QpMbs_rcving
3105601442 User003: event = MBS_FRAME_OK
3105601448 User003: event = MBS_FRAME_OK and Good Request
           INIT : Obj=l_Mbs Source=QpMbs_txing Target=QpMbs_txRun
           ENTRY: Obj=l_Mbs State=QpMbs_txRun
3105601473 ==>Tran: Obj=l_Mbs Sig=MBS_FRAME_OK_SIG Source=QpMbs_rcving New=QpMbs_txRun
           EXIT : Obj=l_Mbs State=QpMbs_txing
```

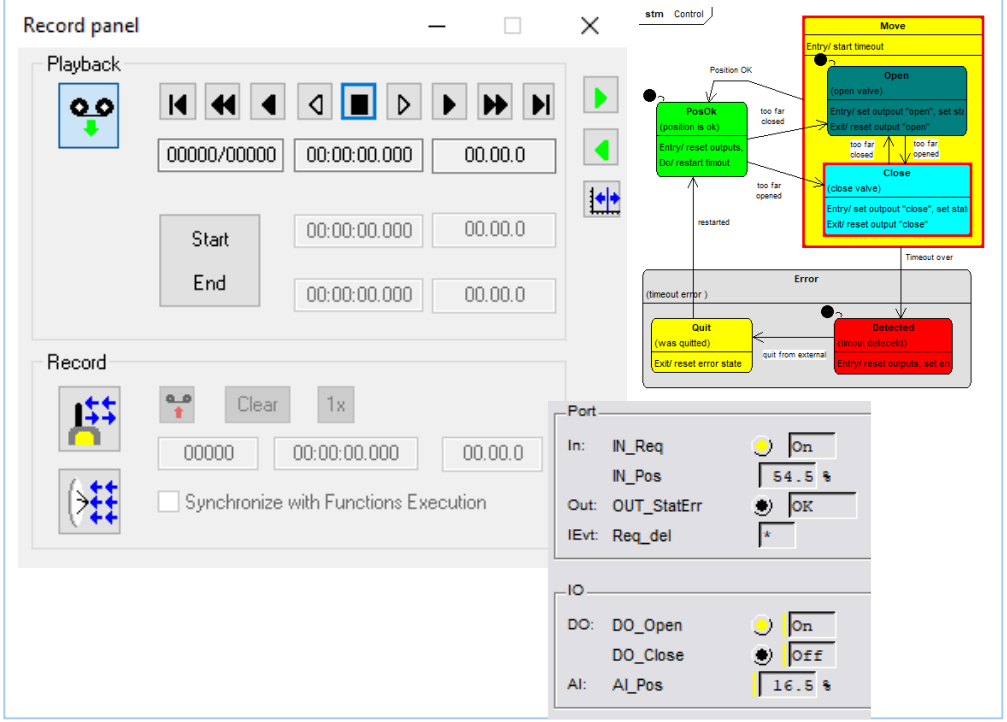
- **Q/Spy**
- **QUTest**
- **QSpyView**

„Gemeinsamkeiten“ im Detail

Debugging

radCASE

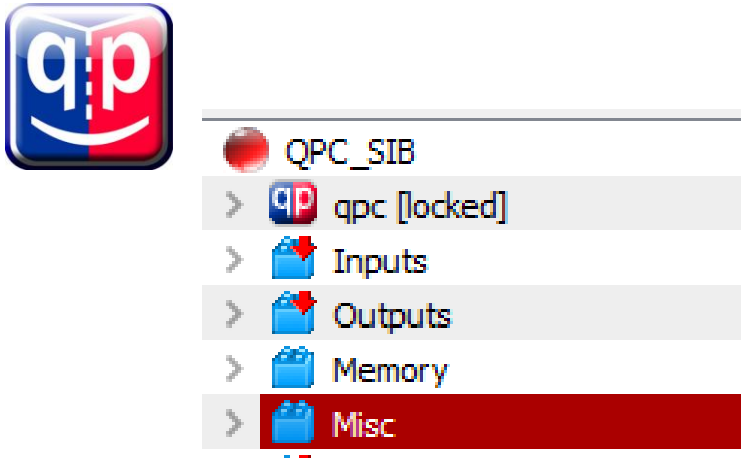
- **animierte Statemachines**
- **record/replay – Funktion**
- **Anzeige aller Variablen**



„Gemeinsamkeiten“ im Detail

Partitionierung

- **Library includes**
- **In jeder Projektdefinition**



- **packages – in Projektdatei**
- **external packages – eigene Datei**

Fazit

- für embedded Systeme
- Generierung von C-Code
- direkt targetfähig

➢ BSP bzw. API wird gebraucht
➢ UML basiert
❖ hierarchische Statemachines

- Objektorientiert
- keine Klassendiagramme
- Objektbaum

❖ Tasking
➢ kooperativ
❖ RTC (Run to completion)
➢ Simulation am PC
➢ Debugunterstützung am PC

- Partitionierung von Projekten
- Projekte in XML-Dateien
- Proprietäre Editoren

Fazit

Fazit

Mit beiden lässt sich nachgewiesenermaßen gut arbeiten

radCASE

©olga meier-sander/pixelio.de

- Sehr komfortabel und bequem
- Viele Extrafeatures
- nicht ganz so schnell

©O. Fischer/pixelio.de

- manchmal etwas unbequem
- einfach, aber sinnvoll ausgestattet
- auf Geschwindigkeit ausgelegt

coming GmbH

Embedded Software seit 1994
…aber nur noch bis Ende 2017

Dipl.-Ing. (FH) Joachim Terasa
coming GmbH
Alte Landstraße 27
85521 Ottobrunn
j.terasa@coming-gmbh.de
http://www.coming-gmbh.de
Telefon +49 89 629890 - 0
Mobil +49 172 6444051
Telefax +49 89 629890 - 89