Konsistenzprüfungen in *OpenPEARL*

Rainer Müller[1] Marcel Schaible[2]

Hochschule Furtwangen, Fakultät IN, Robert-Gerwig-Platz 1, 79120 Furtwangen
mueller@hs-furtwangen.de

Lehrstuhl für Informationstechnik, insb. Realzeitsysteme
FernUniversität, 58084 Hagen
Marcel.Schaible@FernUni-Hagen.de

Workshop Echtzeit 2016 in Boppard

Agenda

1. Motivation
2. Systemteil
3. Probleme
4. Lösung
5. Ergebnis

Motivation

- PEARL ist eine ideale Programmiersprache für Ausbildungszwecke um nebenläufige Abläufe zu beschreiben
- restriktive Programmiersprache
- Konsistenz zur Laufzeit wird durch SIGNALs sichergestellt
- Konsitenz innerhalb des *Problemteils wird* vom Compiler sichergestellt
- Konsistenz innerhalb eines *PEARL-Moduls könnten* durch den Compiler sichergestellt werden
- Konsistenz zwischen Modulen ist eine Schwachstelle

Struktur einer PEARL-Anwendung

- Im Modul kann es SYSTEM- und PROBLEM-Teil geben,
- wobei der Systemteil auch separat stehen kann
- Module exportieren und importieren untereinander *globale* Objekte (Tasks, Prozeduren, Variablen, ...)
- Module können unabhängig voneinander kompiliert werden
- nach der Kompilierung der Module werden diverse Laufzeitbibliothen dazugebunden

Buildvorgang

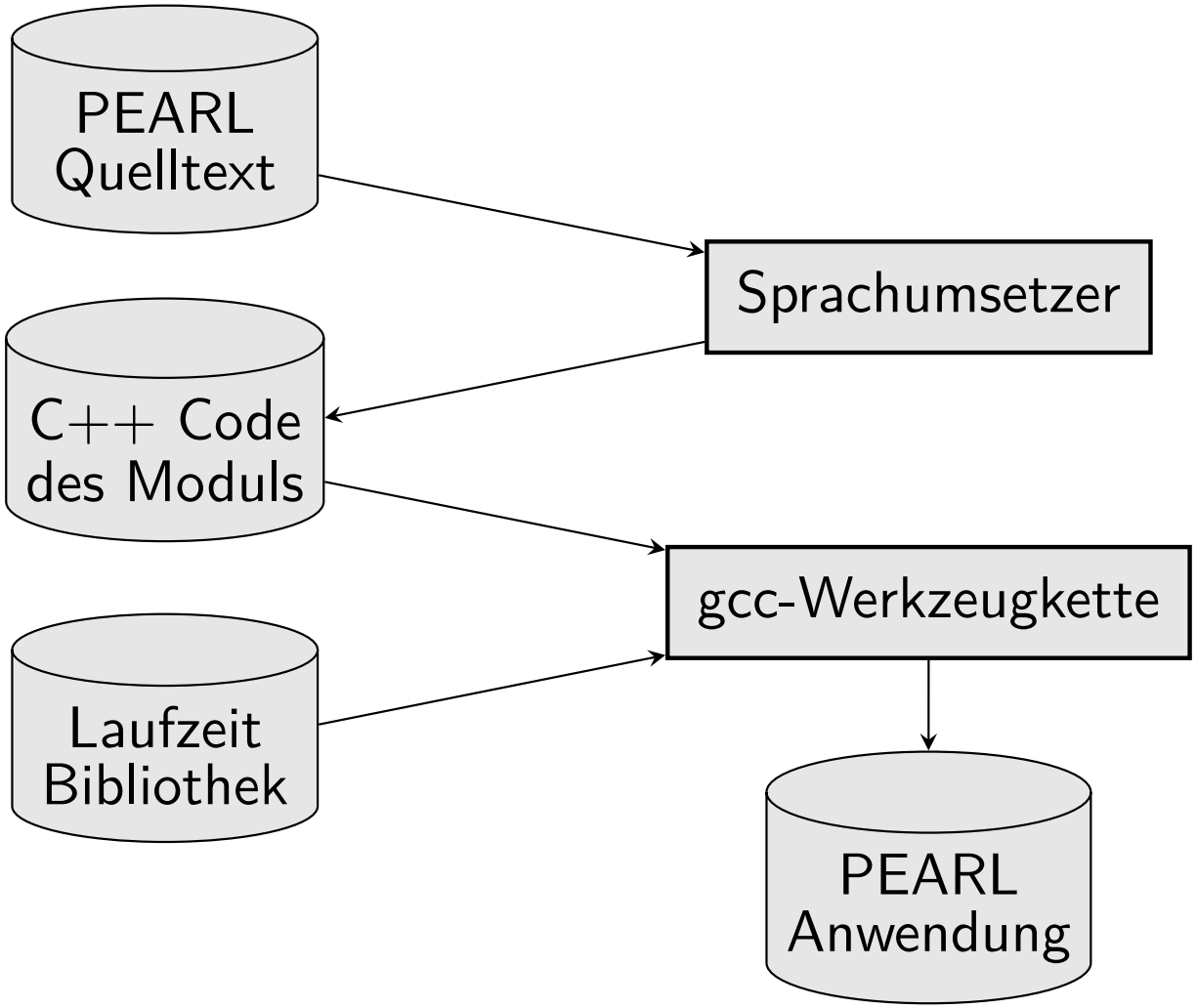


Abbildung: Aufbau des PEARL Übersetzungssystems.

Systemteil I

Beispiel aus PEARL90:

```
Motor: DIGEA(0)*1*1,4;
```

- Benutzerspezifischer Name `Motor`
- wird mit dem Systemgerät `DIGEA` verbunden
- `(0)*1,1,4` beschreibt Adresse der E/A-Komponente, Transferrichtung, Startbit und Zahl der Bits
- Der Systemteil enthält die gesamte plattformspezifische Information

Verbindungsstrukturen, wie im Buch von H. Frevert beschrieben sind in PEARL90 nicht enthalten.

Systemteil II

Für OpenPEARL:

- Syntax mit 4 ganzen Zahlen ist ungenügend, um z.B. Pfadangaben zu realisieren
- Beispiel in OpenPEARL:

```
disc: Disc('/tmp', 10 );
    ! max. 10 Dateien gleichzeitig geoeffnet
    ! im Verzeichnis '/tmp'
```

- keine Beschränkung in Zahl und Typ der Parameter
- allerdings *keine Felder*

Erweiterungen im Systemteil I

- Bussysteme wie I^2C sind weitverbreitet
- Sensoren und Aktoren werden oft über solche Busse angeschlossen
- für z.B. I^2C-Teilnehmer muss es geeignete Darstellung im Systemteil gefunden werden, sodass
  - mehrere Teilnehmer am gleichen Bus
  - und mehrere Busse parallel realisiert werden können.

  $\Rightarrow$ Verbindungspfade wurden wieder eingeführt

```
i2cbus: I2CBus('/dev/i2c-1' , 100000);
Motor: PCF8574Out('20'B4,7,4) --- i2cbus;
Taste: PCF8574In('20'B4, 3,1) --- i2cbus;
```

- Verbindungen besitzen einen Typ, z.B. `I2CProvider`
- der von Clienten (links) geforderte Typ muss vom Provider unterstützt werden

Erweiterungen im Systemteil II

- Konfigurationseinstellung für die Anwendung sollten sichtbar sein
  - gehören in den Systemteil
  - haben jedoch ohne Verbindung zum Problemteil

```
Log('EW') --- StdOut;
```

Probleme

- Definition der Typinformation für Systemnamen

```
SYSTEM;
  disc: Disc('/tmp', 10);
  overflow: FixedOverflowSignal;
  crtl_c: UnixSignal(3);
  Motor: PCF8574Out('20'B4,7,4) ---
         I2CBus('/dev/i2c-1' , 100000);
```

- muß zum Import im Problemteil passen:

```
SPC disc     DATION INOUT SYSTEM ALL GLOBAL;
SPC overflow SIGNAL GLOBAL;
SPC ctrl_c   INTERRUPT GLOBAL;
SPC Motor    DATION OUT SYSTEM BASIC BIT(4) GLOBAL;
```

- speziell für DATIONs müssen Transferdatentyp und Zugriffsrichtung definiert werden

Folgen

- fehlerhafte Benutzung von Systemelementen
  $\Rightarrow$ unzuverlässige Anwendung
- fehlerhafte Nutzung von Anwendungselementen
  $\Rightarrow$ unzuverlässige Anwendung
- unleserliche Fehlermeldungen des Linkers, falls die Symbole für den Linker Typinformation enthalten
  $\Rightarrow$ geringe Akzeptanz bei Programmierern

Notwendigkeit der Kontrolle der Export-/Importschnittstelle

- zwischen System- und Problemteil
- zwischen Problemteilen

Lösungsansätze

- Allmächtiger Sprachumsetzer
  - \+ korrekte Behandlung der Systemeinträge möglich
  - \- Sprachumsetzer muss alle Module kennen
    ⇒ muss alle Module gleichzeitig übersetzen
  - \- Erweiterungen durch neue Systemgeräte beeinflussen den Sprachumsetzter
    ⇒ Instabilitäten im Sprachumsetzer sind zu erwarten
- Separates Tool (**I**nter **M**odule **C**hecker)
  - \+ separate Übersetzbarkeit der Anwendungsmodule bleibt erhalten
  - \+ Information über Systemnamen wird nur dem IMC bekanntgegeben — der Sprachumsetzer ist unabhängig von Zielplattform und Systemgeräten
  - o zusätzlicher externer Übersetzungsschritt notwendig
  - \+ ressourcensparender Betrieb, da erst kurz vor den Linkvorgang die Querbezüge geprüft werden

Build mit IMC I

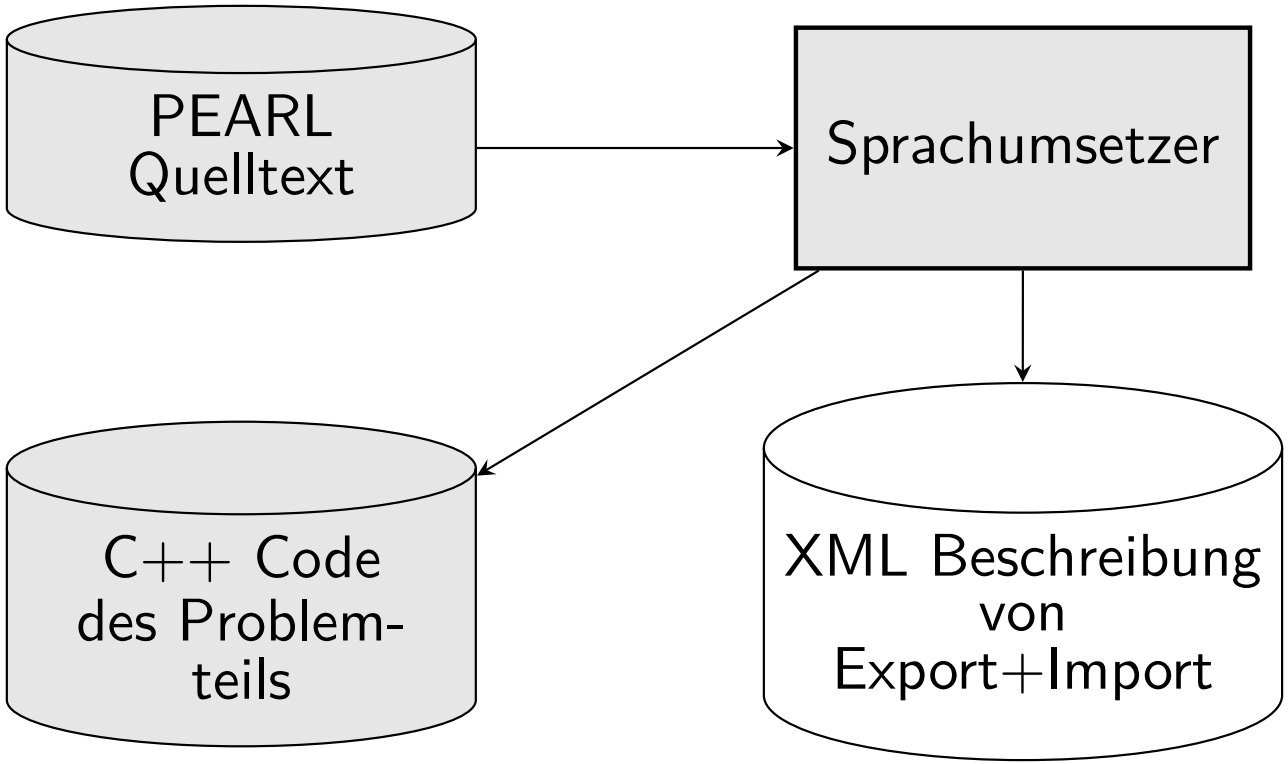
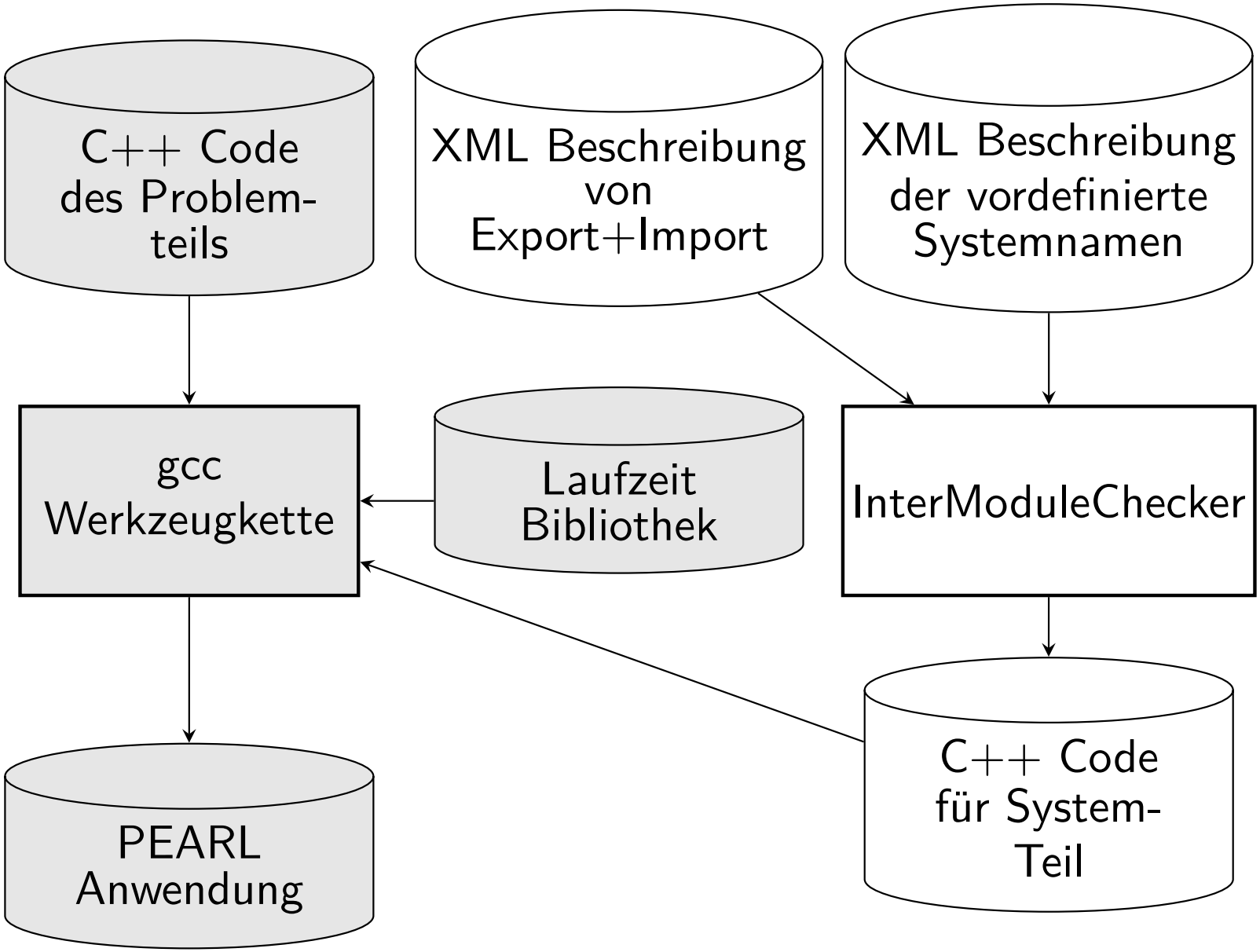


Abbildung: Aufbau des PEARL Übersetzungssystems (1).

Build mit IMC II



Beschreibung der Systemnamen

- XML als Beschreibungssprache wird durch vorhandene Parser gut unterstützt
- für jeden Objekttyp kann eine passende Beschreibung gewählt werden

  SIGNALs werden in *OpenPEARL* in einem Spreadsheet verwaltet.
  $\Rightarrow$ Erzeugung der entsprechenden XML-Strukturen ist generisch möglich

  INTERRUPTs werden wie DATIONs durch Systemmodule verwaltet
  Diese Beschreibungen müssen vom Systementwickler manuell bereitgestellt werden
- Beim Systembuild werden alle benötigten Teile zur gesamten Systembeschreibung zusammengefügt und installiert.

Beispiele für die Systembeschreibung I

`SampleBasicDation` ist eine Vorlage für BASIC Dations.
Es läßt sich der zuletzt geschriebene Wert zurücklesen.

```
<dation name="SampleBasicDation">
   <parameters>
   </parameters>
   <attributes> BASIC, SYSTEM, IN, OUT, INOUT </attributes>
   <data>FIXED(15)</data>
</dation>
```

Beispiele für die Systembeschreibung II

`RPiDigitalIn` realisiert eine Digitaleingabe über GPIO-Pins
Benutzung: `tasten: RPiDigitalIn(25,2,'u');`

```
<dation name="RPiDigitalIn">
   <parameters>
      <FIXED length="31">
         <VALUES>2,3,4,7,8,9,10,11,14,15,17,18,21,22,23,
                 24,25,27</VALUES>
      </FIXED>
      <FIXED length="31" nick="width">
         <FIXEDGT>0</FIXEDGT>
      </FIXED>
      <CHAR length="1"><VALUES>'u','d','n'</CHAR>
   </parameters>
   <attributes>BASIC, SYSTEM, IN </attributes>
   <data>BIT($width)</data>
</dation>
```

Ergebnis der System-Problem-Teil Prüfung

- Parameter können zur Übersetzungszeit weitgehend geprüft werden
- Attribute aus der SPC-Anweisung müssen in der entsprechenden Liste vorhanden sein
- Transferdatentyp des Systemgerätes kann in Abhängigkeit der Parameter formuliert werden

$\Rightarrow$ Systemteil und Problemteil sind konsistent!

Problemteil - Problemteil Konsistenz

- Tasks, Prozeduren und Variablen können per
  - `DCL ... GLOBAL;` exportiert werden
  - `SPC ... GLOBAL;` importiert werden
- Datentypen müssen in jedem Modul neu definiert werden
- ein `INCLUDE`-Mechanismus wird wohl noch eingeführt werden

Lösung im IMC

- der Sprachumsetzer liefert eine Export-/Importbeschreibung auf den elementaren Datentypen
- der IMC vergleicht ...
  - Typen bei Export und Import identisch?
  - fehlender Export?
  - doppelter Export?
  - ungenutzter Export?

Ergebnis

- Namen im Systemteil haben einen bestimmten Typ
- dieser wird mit der SPC-Anweisung im Buildvorgang überprüft
- semantische Überprüfung im Problemteil erfolgt durch den Compiler
- Export und Import zwischen Problemteilelementen werden auch vom IMC überprüft (‚wenn GLOBAL unterstützt wird)
- unzulässige Mehrfachbenutzung von z.B. GPIO-Bits werden spätestens bei der Systeminitialisierung erkannt und die Anwendung noch vor Start der ersten Task beendet.

Werbung!

Mitarbeit erwünscht

Offene Themen:

- Testentwicklung
- Performance Analysen
- ...

Informationen:

Projekt-Webseite	http://OpenPEARL.sourceforge.net
Arbeitskreis	http://www.real-time.de/ak-compiler.html

Ansprechpartner:

AK und Übersetzer	Marcel Schaible	marcel.schaible@fernuni-hagen.de
Laufzeitsystem	Rainer Müller	mueller@hs-furtwangen.de