PEARL für sicherheitsgerichtete Echtzeitprogrammierung

Marcel Schaible Wolfgang A. Halang

Lehrstuhl für Informationstechnik, insb. Realzeitsysteme
FernUniversität, 58084 Hagen

17. - 18. November 2016

Echtzeit 2016: „Internet der Dinge“
in Boppard am Rhein

Agenda

- Motivation
- Sicherheit und Programmverifikation
  - Sicherheitsstufen
  - Programmiermethoden
  - Verifikationsmethoden
- Sicherheitsgerichtete Grundsprache
  - Anforderungen
  - Betrachtung von PEARL unter Sicherheitsaspekten
  - Bildung von Sprachteilmengen
- Teilsprachen für SIL-1 bis SIL-4
- Resüme

Motivation

- Elektronische Systeme für sicherheitskritische Steuer- und Regelanwendungen: Stetig wachsende Komplexität und Kosten
- Wachsendes Sicherheitsbewusstsein
- Forderung: Erstellung von rechnergestützten Systemen mit hinreichendem Grad an Vertrauen in ihre Verlässlichkeit
- Sicherheitskritische Automatisierungsaufgaben: Zulassung durch die Aufsichtbehörden auf der Basis formeller Abnahmen
- Dringender Bedarf an inhärent sicheren Programmiersprachen gemäß den Sicherheitsklassen IEC 61508 für Automatisierungsaufgaben
- Systeme beherrschbar machen und die Ermöglichung der sicherheitstechnischen Abnahme $\rightarrow$ auf höheren Sicherheitsniveaus Beschränkung der Ausdrucksmöglichkeiten

Sicherheit und Programmverifikation

Sicherheitsstufe	**Verifikationsmethode**	**Programmiermethode**
SIL 4	Sozialer Konsens	Ursache-/Wirkungstabelle
SIL 3	Diversitäre Rückwärtsanalyse	Funktionsplan basierend auf verifizierten Bibliotheken
SIL 2	Symbolische Ausführung Formale Korrektheitsbeweise	Prozeduraufruf Zuweisung Alternativauswahl Wiederholungsbegrenzte Schleifen
SIL 1	Alle	Inhärent sicher, statisch, anwendungsorientiert

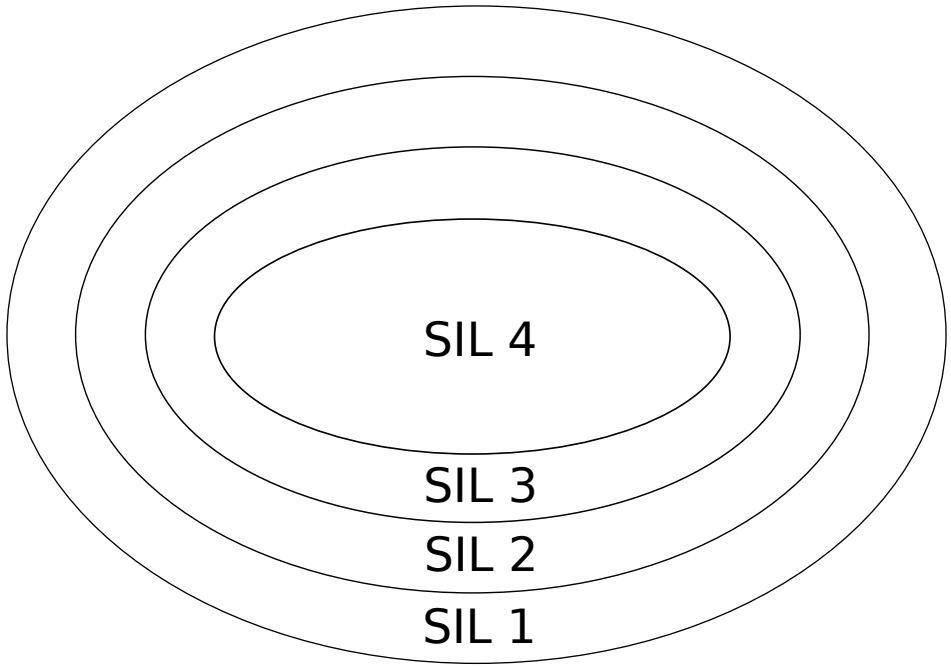
Betrachtung von PEARL unter Sicherheitsaspekten

- Fristen
  - Prioritäten zur Bestimmung der Ausführungsreihenfolge sind abhängig von Zusammenhang, Umgebung und Implementierung → Festlegung von Fertigstellungsterminen
- Fristüberschreitungen
  - Überlast und Fristüberschreitungen → Spezifikation von frühzeitige Behandlung von Ausnahmesituationen
- Zuteilbarkeitsanalyse
  - Bearbeitung einer Reihe von Strukturelementen kann beliebig lange dauern → keine Garantie auf Einhaltung von Zeitbedingungen
- Synchronisierung
  - Semaphore und Bolt fehleranfällig → Einführung funktionaler und strukturierter Synchronisationsmittel
  - Festlegung von maximalen Wartezeiten vor Eintritt in eine kritische Region, der maximalen Aufenthaltszeiten darin sowie geeigneter Ausnahmereaktionen

Sicherheitsgerichtete Grundsprache

- Verklemmungsfreier Schutz von Betriebsmitteln und zeitliche Überwachung von Synchronisierungsoperationen
- Abschätzung von Prozesslaufzeiten und Anwendung zeitgerechter Zuteilungsalgorithmen
- Dynamischer Rekonfiguration verteilter Systeme
- allmähliche Leistungsabsenkung im Fehlerfall
- Handhabung transienter Überlast
- Fehlerentdeckung und Fehlerbehandlung mittels Diversität
- Geschachtelte Teilsprachen
- Wenig sichere Sprachmerkmale werden schrittweise auf höheren Niveaus verboten.

Schachtelung der PEARL-Teilsprachen



PEARL90 mit Mehrrechner-PEARL

Teilsprache SIL 1

- Strukturierte Ausnahmebehandlung mittels TRY-CATCH und RAISE Anweisung
- Betriebsmittelschutz durch die LOCK-PERFORM-UNLOCK Anweisung
- Zeitschranken als obere Schranken für die benötigte Ausführungszeit mittels DUE AFTER und RUNTIME Klausel
- Laufzeitabschätzung von Wiederholungs- oder Iterationsanweisungen durch MAXLOOP-EXCEEDING Klausel
- Dynamische Rekonfiguration verteilter Systeme durch Konfigurationsanweisungen im Initialteil

Teilsprache SIL 2

- Verzicht auf dynamische Konstrukte, die i.A. zu unvorhersehbaren Kapazitäts- und Zeitanforderungen führen
- Begrenzung der Schleifendurchläufe
- Korrektheit lässt sich mit formalen Methoden teilweise werkzeugunterstützt beweisen

SIL 2: Sprachelemente zur Formulierung einer verifizierbaren Teilsprache

⟨*module*⟩ ::=	`MODULE` [(⟨*name*⟩)] [⟨*mod-ext*⟩]; [⟨*intf-spec*⟩]* [⟨*decls*⟩]+ `MODEND`;
⟨*mod-ext*⟩ ::=	`EXTEND` (⟨*name*⟩ [,⟨*name*⟩]+)
⟨*intf-spec*⟩ ::=	`INTERFACE` (⟨*name*⟩) ⟨*intf-decl*⟩* ;
⟨*intf-decl*⟩ ::=	⟨*intf-var-decl*⟩ | ⟨*intf-proc-decl*⟩
⟨*intf-var-decl*⟩ ::=	`SPC` ⟨*name*⟩ : ⟨*type*⟩ `READ`;
⟨*intf-proc-decl*⟩ ::=	`SPC` ⟨*name*⟩ : `PROC` ⟨*lst-of-par*⟩ ⟨*return-type*⟩;
⟨*decls*⟩ ::=	⟨*var-decl*⟩ | ⟨*proc-decl*⟩
⟨*var-decl*⟩ ::=	`DCL` ⟨*name*⟩ : ⟨*type*⟩;
⟨*proc-decl*⟩ ::=	⟨*name*⟩ : `PROC` ⟨*lst-of-par*⟩ ⟨*return-type*⟩; ⟨*body*⟩ `END`;
⟨*lst-of-par*⟩ ::=	(⟨*par-decl*⟩ [, ⟨*par-decl*⟩]*)
⟨*par-decl*⟩ ::=	⟨*name*⟩ : ⟨*type*⟩
⟨*body*⟩ ::=	[⟨*var-decl*⟩]+ [⟨*stmt-seq*⟩]
⟨*stmt-seq*⟩ ::=	[⟨*stmt*⟩;]*
⟨*stmt*⟩ ::=	(⟨*stmt-seq*⟩) | ⟨*assign-stmt*⟩ | ⟨*cond-stmt*⟩ | ⟨*while-stmt*⟩
⟨*assign-stmt*⟩ ::=	⟨*variable*⟩ := ⟨*expr*⟩
⟨*cond-stmt*⟩ ::=	`IF` ⟨*Bool-expr*⟩ `THEN` ⟨*stmt-seq*⟩ [`ELSE` ⟨*stmt-seq*⟩] `FIN`
⟨*while-stmt*⟩ ::=	`WHILE` ⟨*Bool-expr*⟩ `REPEAT` ⟨*stmt-seq*⟩ `END`

Teilsprache SIL 3

- Strukturelemente wie graphische Funktionspläne (FUP) bzw. -blockdiagramme (FBD) gemäß IEC 61131-3:
- Funktions- und Funktionsblockrahmen
- Datenflusslinien
- Namen
- (externe) Anschlusspunkte
- Funktionen speichern keine innere Zustandsinformationen

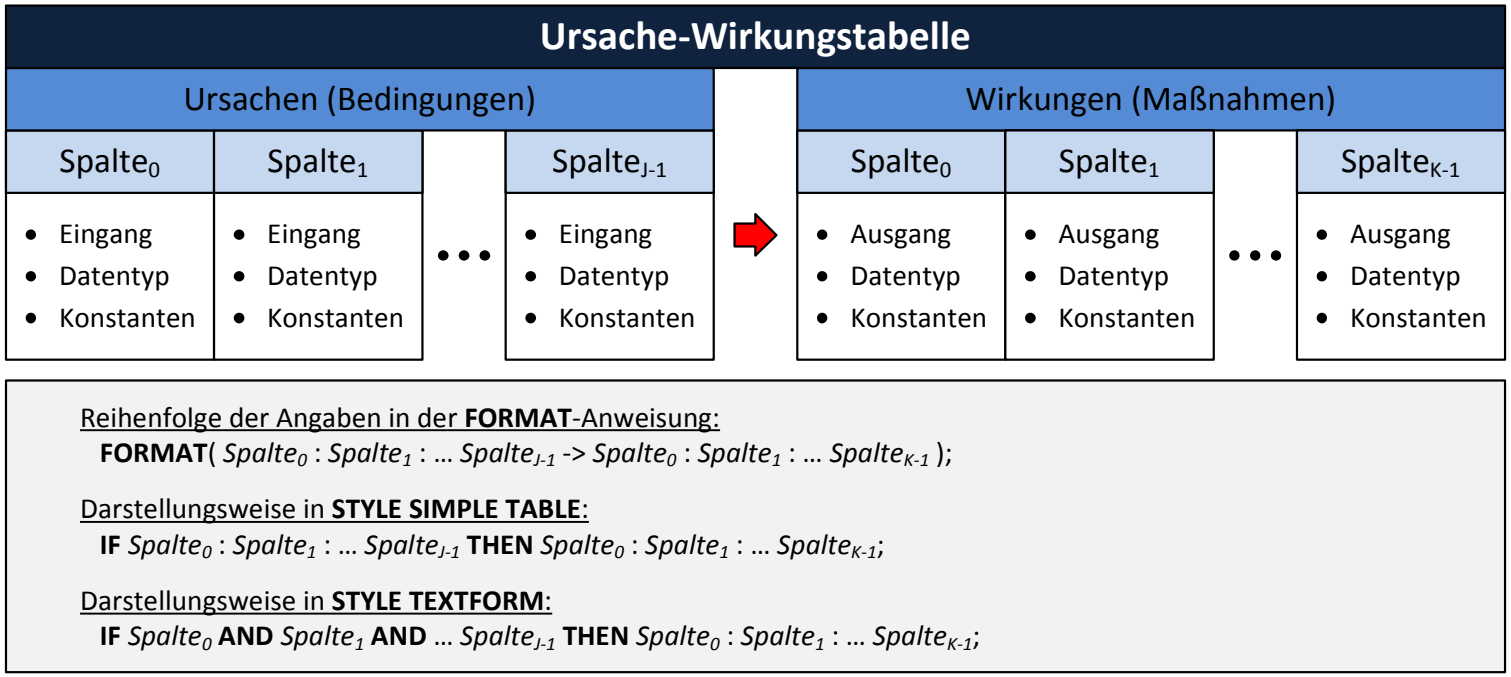
SIL 3: Sprachelemente zur Formulierung sequentieller Ablaufpläne

⟨*sfc::*⟩=	SEQUENCE ⟨*sfc-body*⟩ ENDSEQ;
⟨*sfc-body::*⟩=	⟨*step*⟩ [(⟨*transition*⟩ | ⟨*alternatives*⟩) ⟨*sfc-body*⟩]
⟨*step*⟩::=	STEP [⟨*fbd*⟩] ENDSTEP;
⟨*transition*⟩::=	TRANSITION ⟨*Boolean-expr*⟩;
⟨*alternatives*⟩::=	SELECT [BRANCH ⟨*transition*⟩ ⟨*sfc-body*⟩ ⟨*transition*⟩]+ ENDSCT;

Teilsprache SIL 4

- Anforderungen
  - Präzise und unzweideutige Darstellunsgweise
  - Anschaulich und allgemeinverständlich
  - Verifizierbar und für sozialen Konsens zugänglich
- Ursache-/Wirkungstabellen
  - Zeilen sind mit Ereignissen assoziiert
  - Ereignisse bewirken logische Vorbedingungen

Teilsprache SIL 4



Quelle: Jürgen Hillebrand, Definition einer sicherheitsgerichteten Echtzeitprogrammiersprache. Masterarbeit, FernUniversität Hagen, August 2013.

SIL 4: Sprachelemente zur Formulierung von Ursache-/Wirkungstabellen

⟨*CET*⟩::=	`CETABLE` [⟨*BezeichnerUWT*⟩] [⟨*MaximaleLaufzeit*⟩] [⟨*VereinbarungSIL*⟩] `";"` [[⟨*StatementSequenzUWT*⟩] [⟨*Programmbloecke*⟩]]* `END` [⟨*BezeichnerUWT*⟩ `|"("` ⟨*BezeichnerUWT*⟩ `")"`]`";"`
⟨*CauseEffect*⟩::=	`SETEFFECT` {`"("`⟨*Funktionsprozedurangabe*⟩`")"|` ⟨*Funktionsprozedurangabe*⟩} `TO` ⟨*Cause*⟩ `FIN ";"`
⟨*Cause*⟩::=	`CAUSE "("`⟨*Cond*⟩`")" ";"` [`OR CAUSE "("`⟨*Cond*⟩`")" ";"`]*
⟨*Cond*⟩::=	⟨*BoolSglExprCause*⟩ [`AND` ⟨*BoolSglExprCause*⟩]*
⟨*BoolSglExprCause*⟩::=	[`NOT`] ⟨*Op1*⟩[{`EQ | " ==")`}⟨*Op2*⟩]

Resüme

- PEARL
  - Geschachtelte Teilmengen für die vier Sicherheitsintegritätsniveaus nach IEC 61508
  - Vereinigt sämtliche bekannten Sprachmittel zur Förderung funktionaler Sicherheit
  - Orientiert sich an der *menschlichen Verständnisfähigkeit*
- Programmverifikation: Sozialer Prozess zur Erreichung eines Konsenses
- Spezifikationsebene: Ausfüllen von Entscheidungstabellen $\rightarrow$ in allen Aspekten äußerste Einfachheit
- Prüfung von in dieser Sprache erstellter Software kann von den Technischen Überwachungsvereinen mit größerer Vertrauenswürdigkeit und vertretbarem Aufwand durchgeführt werden.
- Inhärent sichere Sprachkonstrukte $\rightarrow$ Senkung des Risikos für Menschenleben, Umwelt und Anlagen als auch der Wartungskosten

Vielen Dank für Ihre Aufmerksamkeit!

Fragen?