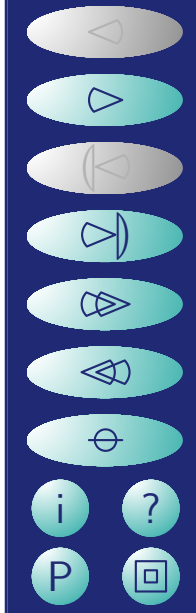
Von Algol 68 zu SafePEARL

Wolfgang A. Halang und **Marcel Schaible**
{wolfgang.halang|marcel.schaible}@fernuni-hagen.de

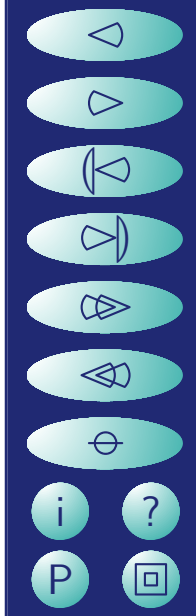


Entwicklung von Programmiersprachen

Jahr	Programmiersprache	Entwickler	Beeinflusst von
1945	Plankalkül	Zuse	–
1953	*Fortran*	Backus	–
1957	Fortran	Backus	A-2
1958	Algol 58	ACM/GAMM-Mitglieder	–
1958	Fortran II	IBM	Fortran
1960	Algol 60	Backus, Naur	Algol 58
1964	PL/I	IBM	Fortran, Algol 60
1965	Fortran IV	IBM/Fortran Working Group	Fortran II
1966	Algol W	Wirth	Algol 60
1966	Fortran 66	ANSI-Mitglieder	Fortran IV
1967	*Simula*	Dahl, Nygaard	Algol 60
1968	*Algol 68*	van Wijngaarden, Koster et al.	Algol 60
1970	PEARL	AEG, BBC, Siemens, GfK Karlsruhe	Algol 60, PL/I
1971	Pascal	Wirth, Jensen	Algol 58
1972	C	Ritchie	B, BCPL, Algol 60
1977	Basic PEARL	PEARL-Verein	PEARL
1978	Fortran 77	ANSI-Mitglieder	Fortran IV
1978	Modula-2	Wirth	Pascal
1980	Ada	Ichbiah, Honeywell Bull	–
1982	Full PEARL	PEARL-Verein	Basic PEARL
1983	C++	Stroustrup	C, Simula 67, Algol 68
1983	Ada 83	DoD/ANSI-Mitglieder	Ada

Entwicklung von Programmiersprachen

1988	Oberon	Wirth	Modula-2
1989	ANSI C (C89)	ANSI-Mitglieder	C, Algol 68
1989	Mehrrechner-PEARL	Steusloff	Bsic, Full PEARL
1991	Fortran 90	ANSI-Mitglieder	Fortran 77
1995	Ada 95	ISO/ANSI-Mitglieder	Ada 83
1997	Fortran 95	ISO/Fortran Working Group	Fortran 90
1998	ISO C++ 98	ISO Working Group	C++
1998	PEARL90	GI/GMA/ITG FA Echtzeitsysteme	Full PEARL
1999	ISO C 99	ISO Working Group	ISO C 95
2003	ISO C++ 2003	ISO Working Group	ISO C++ 98
2004	Fortran 2003	ISO/Fortran Working Group	Fortran 95
2007	Ada 2005	Ada Rapporteur Group	Ada 95
2011	ISO C++ 2011	ISO Working Group	ISO C++ 2003
2012	OpenPEARL	GI/GMA/ITG FA Echtzeitsysteme	PEARL90
2017	ISO C++ 2017	ISO Working Group	ISO C++ 2011
2018	SafePEARL	GI/GMA/ITG FA Echtzeitsysteme	PEARL90 und OpenPEARL



Zwei Zitate

E.W. Dijkstra, 1989 It is time to unmask the computing community as a Secret Society for the Creation and Preservation of Artificial Complexity.

K.H. Biedenkopf, 1994 Fortschritt ist der Weg von der primitiven über die komplizierte zur einfachen Lösung.

Nota bene: Komplexität $\neq$ Kompliziertheit;
Dijkstra meint „complicatedness“.

Echtzeitprogrammiersprachen

Klassifikation maschinenunabhängiger Hochsprachen für die Prozessautomation:

- Anwendungsspezifisch (problemorientiert)

  Sprachen: EXAPT, ATLAS, DIPOL, PSF, ...

  Parameterisierbare Pakete: MADAM, ARSI, BICEPS, CONSUL, AML/7

- Universal (prozedural)

  Spracherweiterungen:

  FORTRAN, BASIC, PL/1, ...

  Neue Sprachen ohne Echtzeitkonstrukte:

  CORAL 66, POLYP, RTL/2

  Neue Echtzeitsprachen:

  PEARL, LTR (PROCOL), Ada, IEC 61131-3

In der Automatisierungstechnik werden unzulängliche Sprachen wie Assembler, C oder C++ jedoch (viel) häufiger eingesetzt als originäre Echtzeitsprachen.

Geschichte von PEARL

Das Akronym steht für **P**rocess and **E**xperiment **A**utomation **R**ealtime **L**anguage.

1969 Beginn der Entwicklung

1973 Erste Version, Pilotimplementierungen und -erprobungen

1979 Norm DIN 66 253 Teil 1 „Basic PEARL“

1981 Norm DIN 66 253 Teil 2 „Full PEARL“

1989 Norm DIN 66 253 Teil 3 „Mehrrechner PEARL“

1998 Norm DIN 66 253-2 „PEARL 90“

2018 Norm DIN 66 253 „SafePEARL“

Mehrrechner-PEARL

Die einzige normierte Programmiersprache für verteilte Systeme erlaubt

- Architektur- und Konfigurationsbeschreibungen sowie Parameterisierungen – ohne ausführbaren Code zu erzeugen – von Übersetzern, Bindern und Ladern
- Nachrichtenübermittlung zur asynchronen und zur zeitlich überwachten synchronen Kommunikation zwischen Knoten,
- dynamische Rekonfiguration zur Unterstützung von Reparatur, Wartung und allmählicher Leistungreduktion.

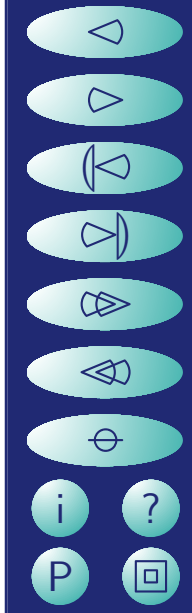
Dynamische Rekonfiguration

CONFIGURATION
    initial-part
     [reconfiguration-part-string]
ENDCONFIG;

initial-part::=load-clause-string
reconfiguration-part::=
STATE Boolean-expression
                remove-clause-string
                load-clause-string
            ENDRECONF;
load-clause::=
            LOAD collection-identifier TO processor-identifier;
remove-clause::=
            REMOVE collection-identifier FROM processor-identifier;

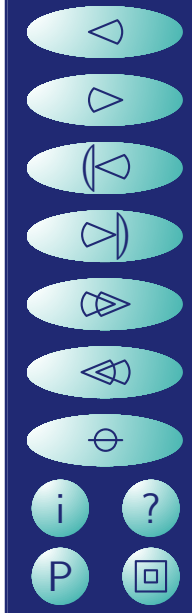
Dynamische Rekonfiguration

```
CONFIGURATION;
  /* Initial Part */
  COLLECTION    DrillControl
     MODULES    AdvanceCrtl, SleighCrtl, RPMCrtl
     PORTS      PDrill1, PDrill2, PDrill3;

  COLLECTION    OperationSupervision
     MODULES    DrillDataEval, Statistics, DataLog
     PORTS      POpSv1, POpSv2, POpSv3, POpSv4;

  LOAD          DrillControl TO Station1;
  LOAD          OperationSupervision TO Station2;
  CONNECT       DrillControl.PDrill1 -> OperationSupervision.POpSv1;
  CONNECT       DrillControl.PDrill2 -> OperationSupervision.POpSv2
                PREFER (TerminalPoint1A,TerminalPoint1B);
```

Dynamische Rekonfiguration

```
  /* Reconfiguration Part */
  STATE          (Station2.down AND NOT Station3.down)
  BEGIN
    DISCONNECT  DrillControl.PDrill1 -> OperationSupervision.POpSv1;
    DISCONNECT  DrillControl.PDrill2 -> OperationSupervision.POpSv2
    REMOVE      OperationSupervision FROM Station2;
    LOAD        OperationSupervision TO Station3;
    CONNECT     DrillControl.PDrill1 -> OperationSupervision.POpSv1;
                VIA (TerminalPoint2A,TerminalPoint2B);
    CONNECT     DrillControl.PDrill2 -> OperationSupervision.POpSv2
                VIA (TerminalPoint2A,TerminalPoint2B);
  END
CONFEND;
```

IEC 61131-3

Die Sprachfamilie für speicherprogrammierbare Steuerungen nach IEC 61131-3 besteht aus

Anweisungsliste Assembler für einen hypothetischen Prozessor

Kontaktplan Schaltplan auf Bauelementeniveau

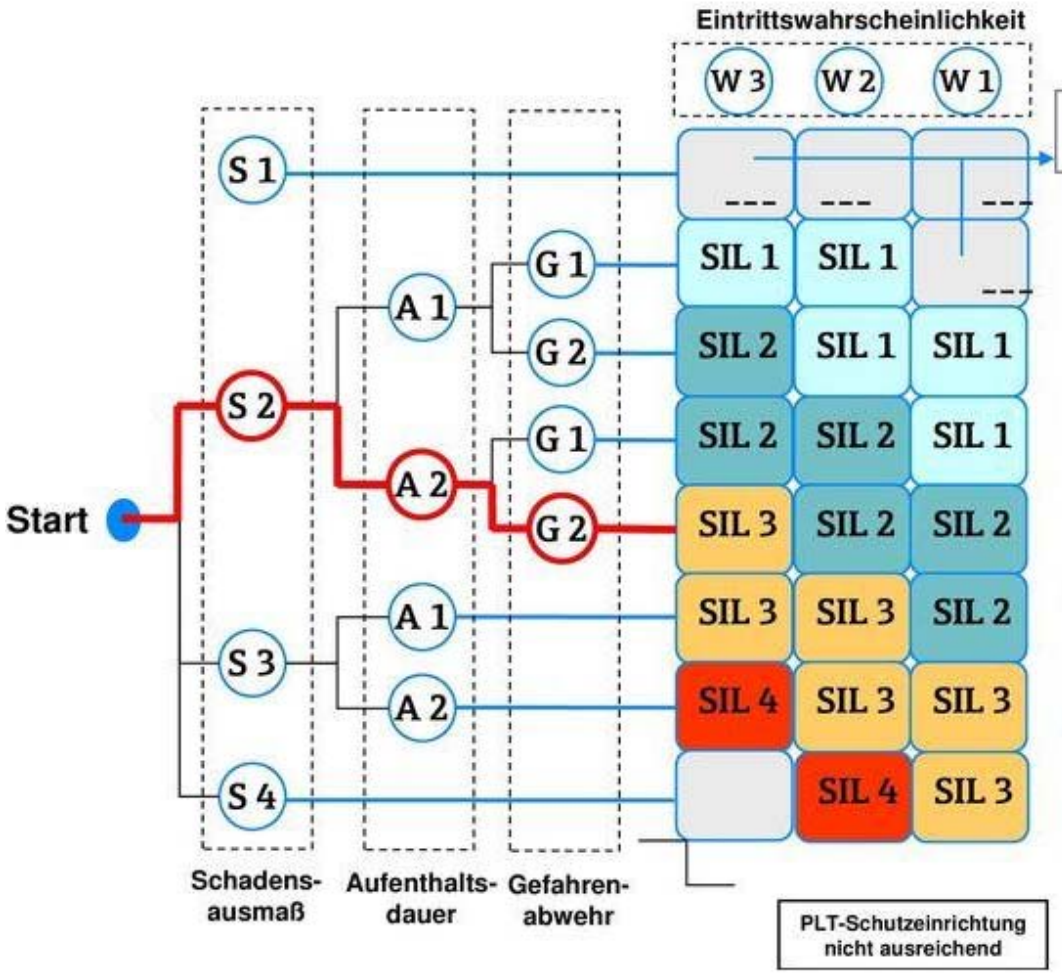
Strukturierter Text Pascalähnliche Hochsprache

Funktionsplan Struktur-/Schaltplan auf Komponentenniveau

Sequentieller Ablaufplan Textuell oder graphisch beschriebene Zusammenstellungen von Schritten, Transitionen und Aktionen

Trotz fehlender Echtzeitkonstrukte sind sie wegen der Betriebsart von SPSen als Echtzeitsprachen zu betrachten.

Sicherheitsanforderungsklassen



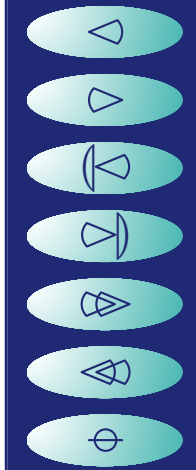
Schadenausmaß
S1 leichte Verletzung einer Person oder kleinere schädliche Umwelteinflüsse, die z.B. nicht unter die Störfallverordnung fallen.
S2 schwere, irreversible Verletzung einer oder mehrerer Personen oder Tod einer Person oder vorübergehende größere schädliche Umwelteinflüsse, z.B. nach Störfallverordnung.
S3 Tod mehrerer Personen oder lang andauernde größere schädliche Umwelteinflüsse, z.B. nach Störfallverordnung.
S4 katastrophale Auswirkung, sehr viele Tote.

Aufenthaltsdauer
A1 selten bis öfter
A2 häufig bis dauernd

Gefahrenabwehr
G1 möglich unter bestimmten Bedingungen
G2 kaum möglich

Eintrittswahrscheinlichkeit
W1 sehr gering
W2 gering
W3 relativ hoch

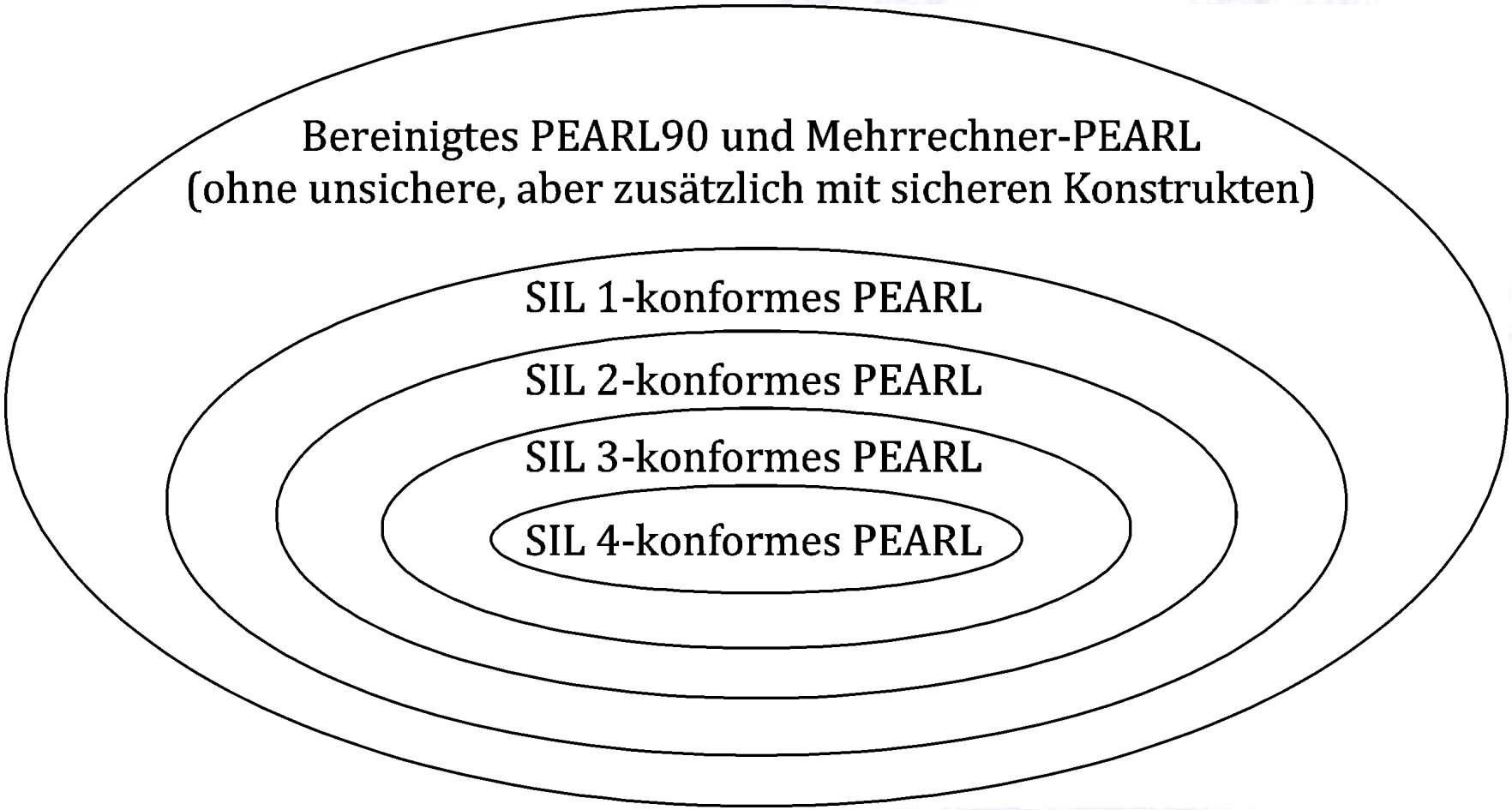
Genau andersherum



Programmverifizierbarkeit

Sicherheits-stufe	**Verifikationsmethoden**	**Vertrauens-würdigkeit**	**Teilsprachenkonstrukte**
SIL 4	Sozialer Konsens	hoch	Ursache-/Wirkungstabelle
SIL 3	Diversitäre Rückwärtsanalyse	recht hoch	Funktionsplan basierend auf verifizierten Bibliotheken
SIL 2	Symbolische Ausführung Formale Korrektheitsbeweise Hoare-Kalkül	mittel	Prozeduraufruf Zuweisung Alternativenauswahl Wiederholungsbegrenzte Schleife
SIL 1	Alle	eingeschränkt	Inhärent sicher, statisch, anwendungsorientiert

Teilmengen von SafePEARL



Umfang von SafePEARL

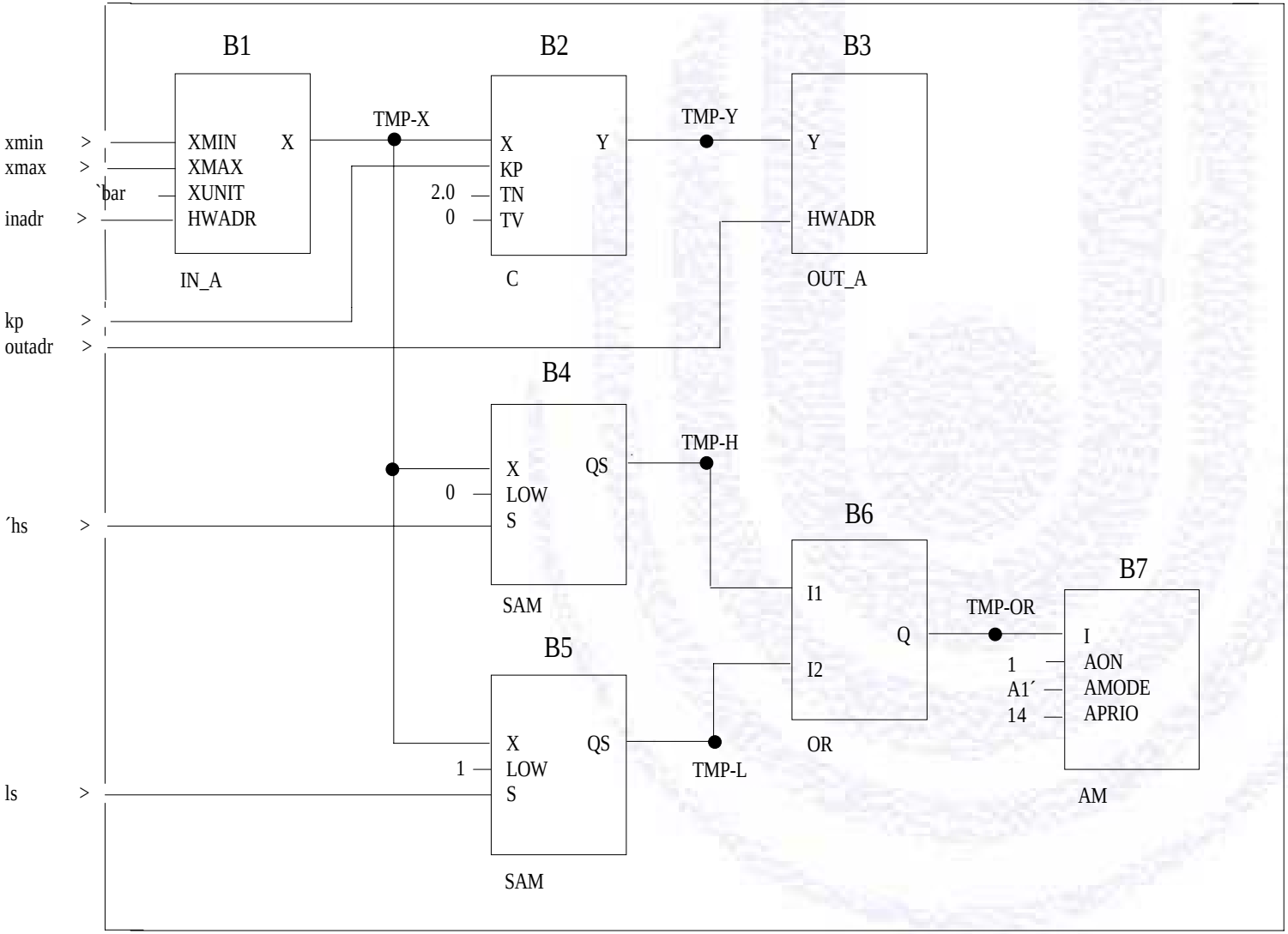
Die Norm DIN 66 253 „SafePEARL“ umfasst 136 Textseiten bzw.

genau eine:

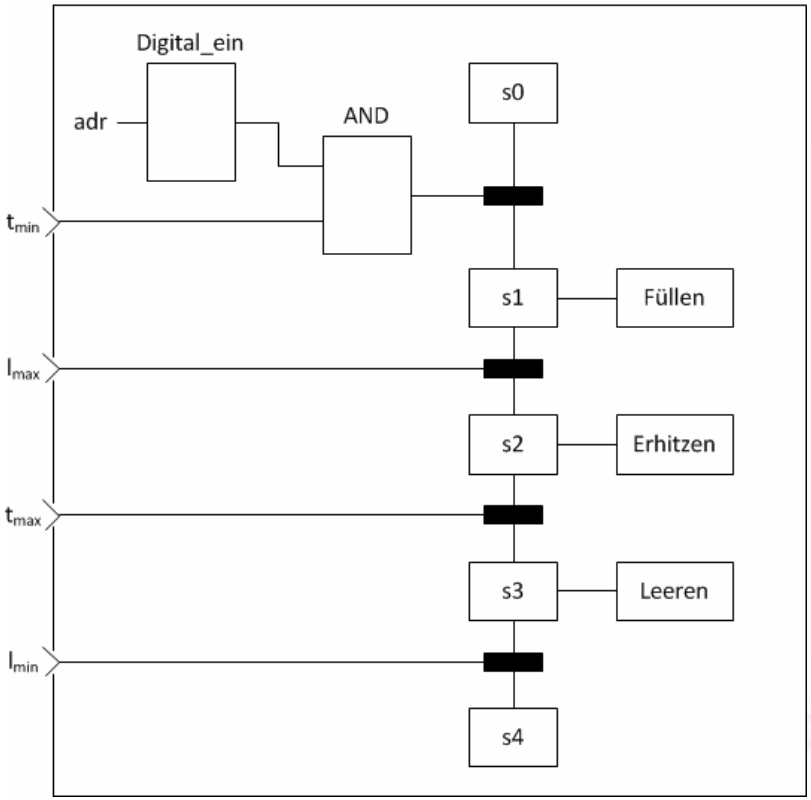
Sicherheitsgerichtete Sprachteilmengen

Anweisung/Klausel	SIL0	SIL1	SIL2	SIL3	SIL4
GOTO, EXIT	+	–	–	–	–
(bedingte) Ausdrücke und Zuweisungen	+	+	+	–	–
bedingte Anweisungen und Anweisungsauswahl	+	+	+	–	–
Angabe physikalischer Einheiten	+	+	+	+	+
Ursache-Wirkungstabellen	+	+	+	+	+
sequentielle Ablaufpläne	+	+	+	+	–
Synchronisierung mit SEMA- und BOLT-Variablen	+	–	–	–	–
Synchronisierung mit LOCK und TIMEOUT-Klausel	+	+	+	–	–
Verwendung interner Signale	+	+	+	–	–
Verwendung von Unterbrechungssignalen	+	+	+	–	–
Verwendung von Tasks	+	+	+	–	–
mit Prioritäten	+	+	–	–	–
mit Fristenangaben und Zeitüberwachung	+	+	+	–	–
(Funktions-) Prozeduraufrufe	+	+	+	+	–
Wiederholungen	+	+	–	–	–
mit MAXLOOP-Klausel	+	+	+	–	–
Verwendung von Zeigern und Referenzen	+	–	–	–	–
PUT/GET, WRITE/READ, CONVERT	+	+	+	–	–
TAKE/SEND	+	+	+	+	+
verteilte Systeme	+	+	+	+	+
dynamische Rekonfiguration	+	+	+	+	–
Botschaftenaustausch	+	+	+	–	–

Funktionspläne

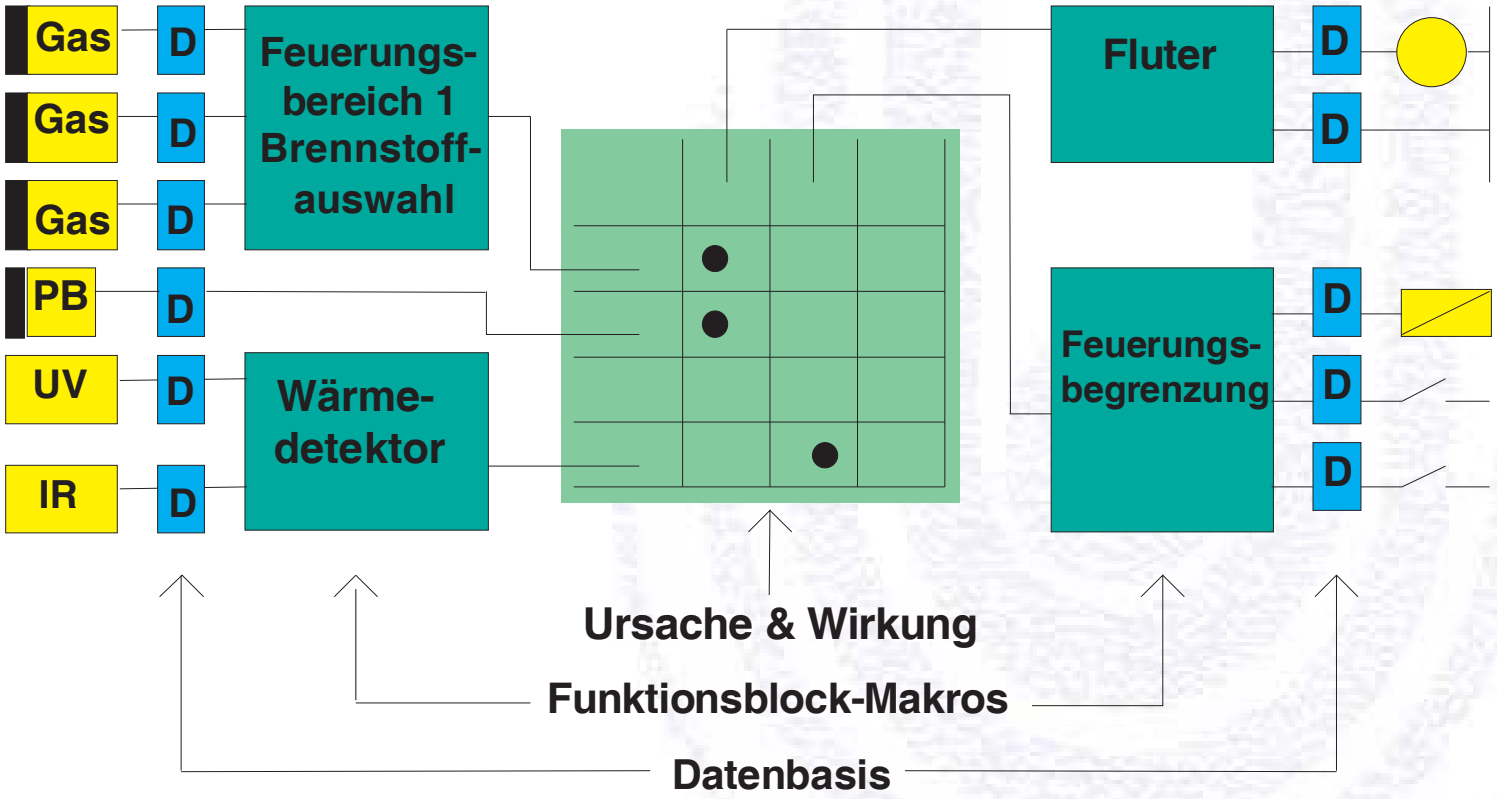


Ablaufplan einer Behältersteuerung




```
SEQUENCE
STEP ENDSTEP;
TRANSITION Digital_ein(adr)
    AND Temperatur LE tmin;
STEP Call Fuellen ENDSTEP;
TRANSITION Fuellstand GE lmax;
STEP Call Erhitzen ENDSTEP;
TRANSITION Temperatur GE tmax;
STEP Call Leeren ENDSTEP;
TRANSITION Fuellstand LE lmin;
STEP ENDSTEP;
ENDSEQ;
```

Ursache-Wirkungstabellen

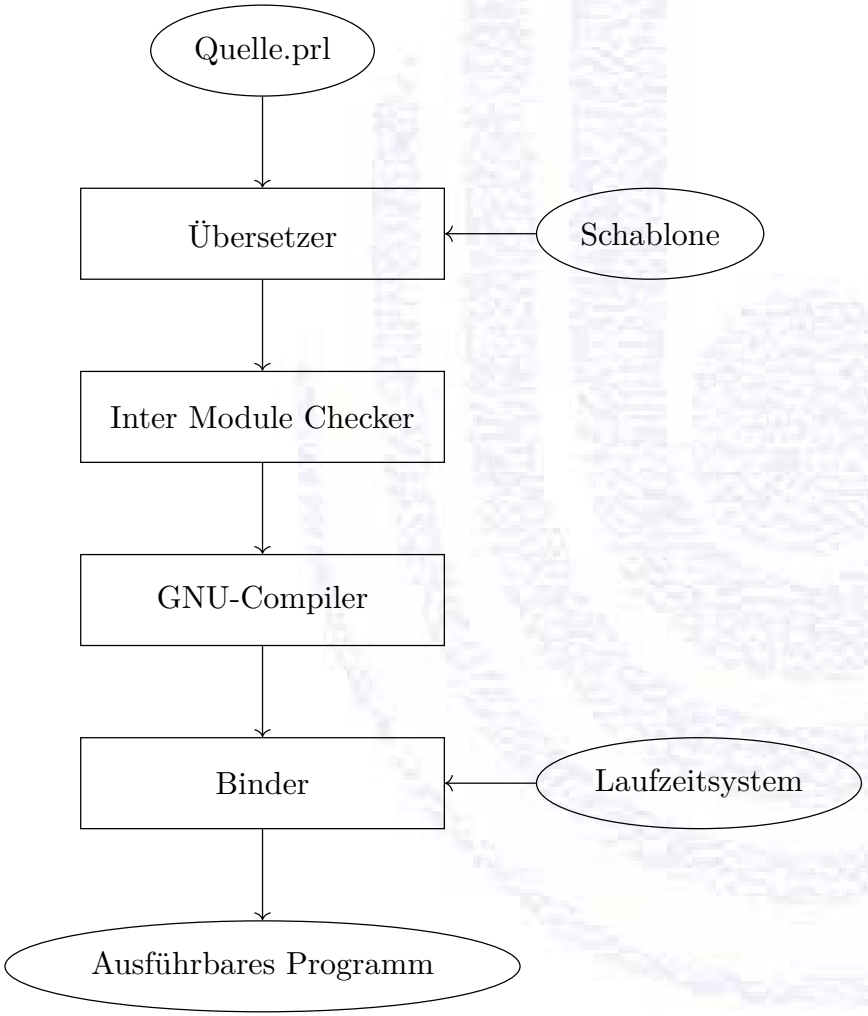


Ursache-Wirkungstabellen sind verfügbar zur Konfigurierung proprietärer Prozessleitsysteme (z.B. von Triconex und ABB).

Ursache-Wirkungstabelle in SafePEARL

```
MODULE(Ursache_Wirkungstabelle) SAFEGUARD SIL4;
SYSTEM;
SPECIFY Drucksensor, Fuellhoehe, Thermoelement DATION IN SYSTEM BASIC;
SPECIFY Ventil DATION OUT SYSTEM BASIC;
PROBLEM;
CETABLE Kessel;
DECLARE Druck, Fuellhoehe, Temperatur FIXED;
TAKE Druck      FROM Drucksensor;
TAKE Fuellhoehe FROM Fuellstand;
TAKE Temperatur FROM Thermoelement;
CAUSE Druck LE 5 AND Fuellhoehe LE 85 AND Temperatur LE 44 EFFECT SEND '0'B1 TO Ventil;
CAUSE Druck LE 5 AND Fuellhoehe LE 85 AND Temperatur GT 44 EFFECT SEND '0'B1 TO Ventil;
CAUSE Druck LE 5 AND Fuellhoehe GT 85 AND Temperatur LE 44 EFFECT SEND '1'B1 TO Ventil;
CAUSE Druck LE 5 AND Fuellhoehe GT 85 AND Temperatur GT 44 EFFECT SEND '0'B1 TO Ventil;
CAUSE Druck GT 5 AND Fuellhoehe LE 85 AND Temperatur LE 44 EFFECT SEND '0'B1 TO Ventil;
CAUSE Druck GT 5 AND Fuellhoehe LE 85 AND Temperatur GT 44 EFFECT SEND '1'B1 TO Ventil;
CAUSE Druck GT 5 AND Fuellhoehe GT 85 AND Temperatur LE 44 EFFECT SEND '1'B1 TO Ventil;
CAUSE Druck GT 5 AND Fuellhoehe GT 85 AND Temperatur GT 44 EFFECT SEND '1'B1 TO Ventil;
END Kessel;
MODEND Ursache_Wirkungstabelle;
```

OpenPEARL-Übersetzer



Vergleich OpenPEARL mit PEARL90

- Fehlerhafte Systemzustände wie FIXED-Variablen-Überläufe werden mittels Signalen zum Laufzeitsystem weitergeleitet.
- Operatorneudefinition mittels `OPERATOR` ist nicht mehr möglich.
- Generischer Datentyp `VOID` und programmatische Typumwandlung mittels `BY TYPE` werden nicht unterstützt.
- Verschachtelte Prozeduren sind nicht mehr zugelassen.
- Bedingte Ausdrücke in Zuweisungen sind weggefallen.
- `BIT(n)`-Werte sind hin und her konvertierbar in `FIXED(n-1)`-Werte.
- Für Datentyp `FIXED` sind Wortlängen von 0 bis 63 Bits erlaubt.
- Der Datentyp `FLOAT` kann Mantissenlängen von 24 oder 53 Bits haben.

Entwicklungsstand von OpenPEARL

Der Übersetzer behandelt z.Zt. alle PEARL90-Sprachkonstrukte außer:

- CONVERT, STRUCT und TYPE
- INTERRUPT und SIGNAL

Das Laufzeitsystem

- ist bis auf kleinere Erweiterungen stabil,
- in C++ implementiert,
- arbeitet weitgehend mit Templates und
- läuft auf den meisten gängigen Linux-Distributionen.

Unterstützte Plattformen: x86, Raspberry Pi, LPC1768 und ESP32.

Ausblick

- Fehlende Sprachkonstrukte implementieren
- Testabdeckung verbessern
- Tasks bei Konsoleingaben am ESP32 adressierbar machen
- Eingebetteten Webserver um Benutzerschnittstelle erweitern
- GCC durch CLANG/LLVM ersetzen
- LLVM-basierten Codegenerator erstellen
- OpenPEARL-Lexer und -Parser für LLVM erstellen
- OpenPEARL-Übersetzer sukzessive um SafePEARL-Konstrukte erweitern
- Der Einsatz von SafePEARL könnte wesentlich zum Erfolg der Initiative Industrie 4.0 beitragen.