Verifikation einer Funktionsblockbibliothek für die Prozessautomatisierung

Marc L. Schulz

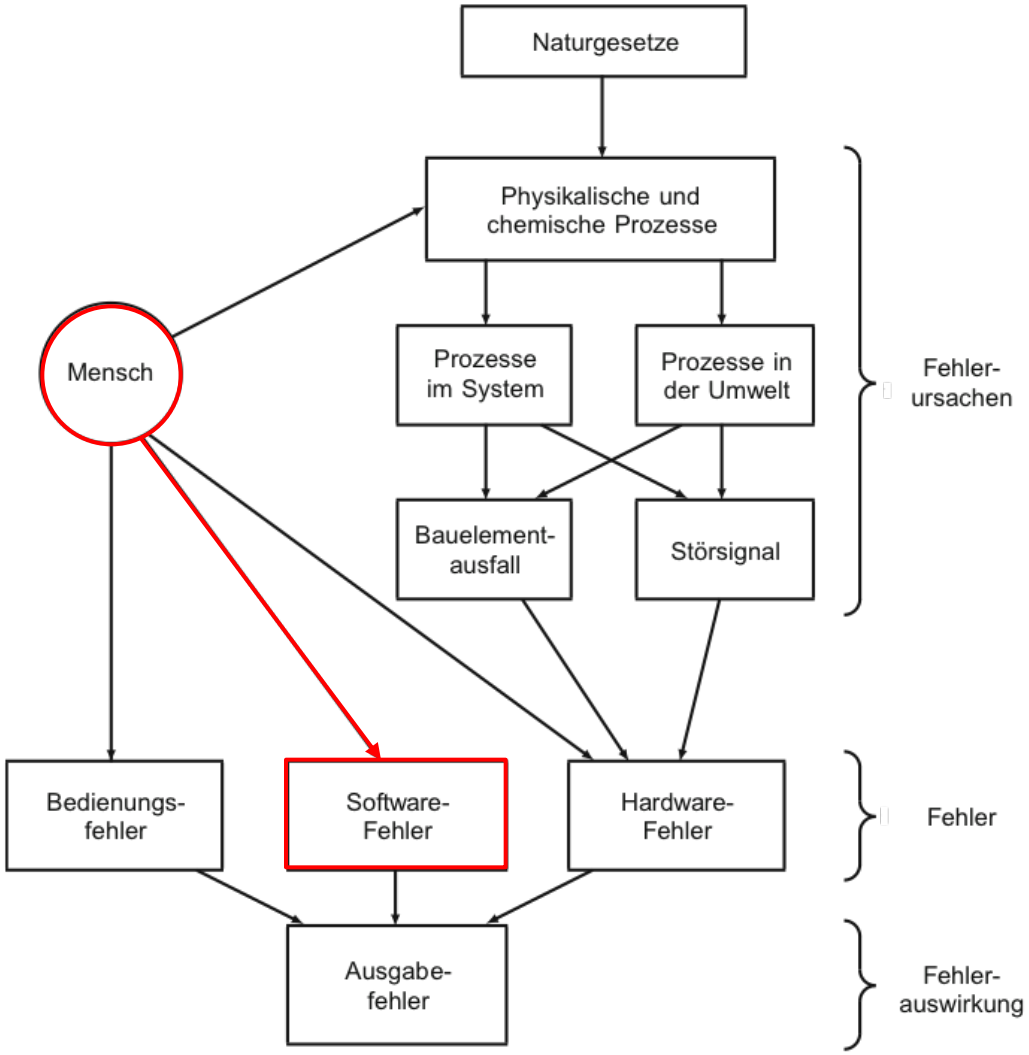
Vortrag auf der GI/GMA/ITG-Tagung Echtzeit 2019

Boppard, 22. November 2019

1. **MOTIVATION** der Forschungsarbeit
2. **DATENVERARBEITUNGSGANLAGE** zur Realisierung vollständig verifizierbarer Automatisierungssysteme
3. **QUASIEMPIRISCHER BEWEIS** als Verifikationsstrategie
4. **VERIFIKATION** einer Funktionsblockbibliothek für die Prozessautomatisierung

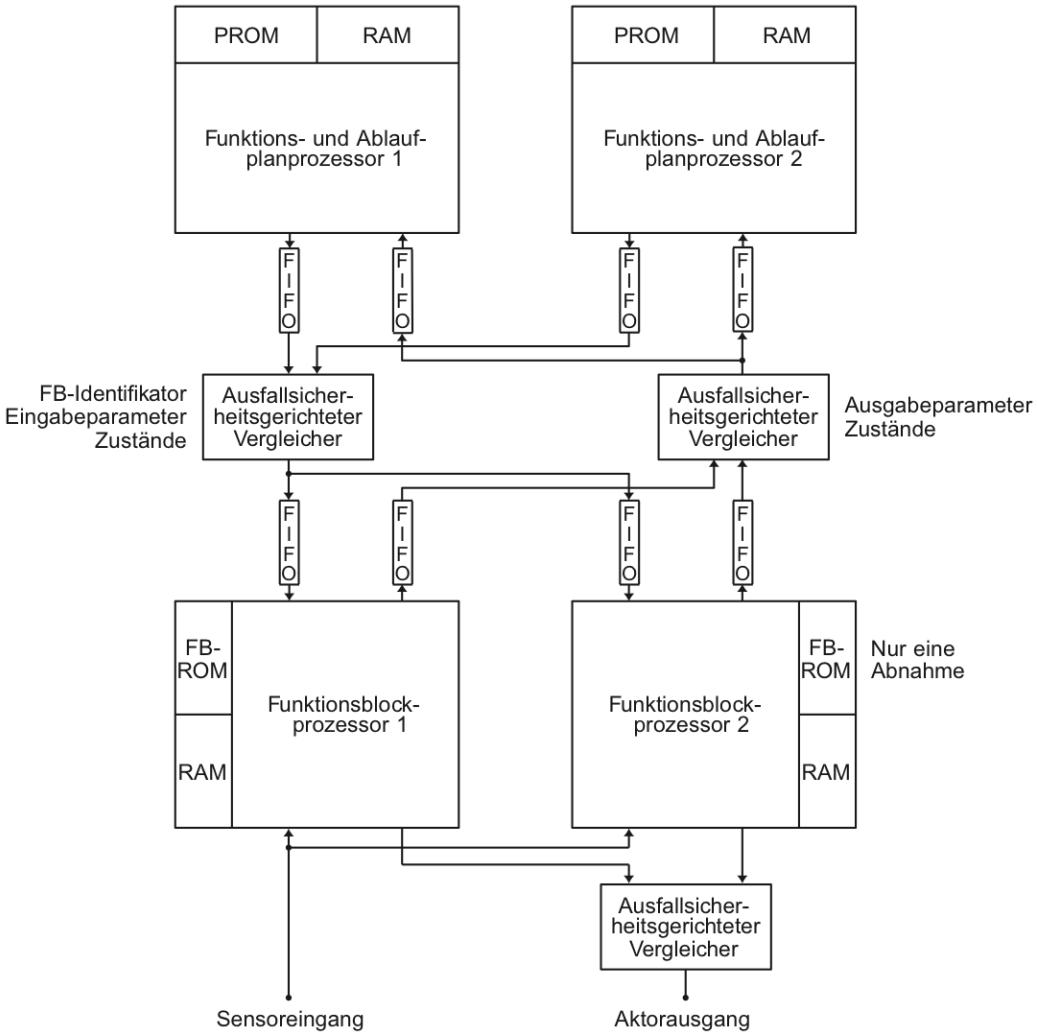
Verlässliche Software für sicherheitsgerichtete Automatisierungssysteme

Softwarefehler sind immer vom Menschen verursacht und somit systematischer Natur



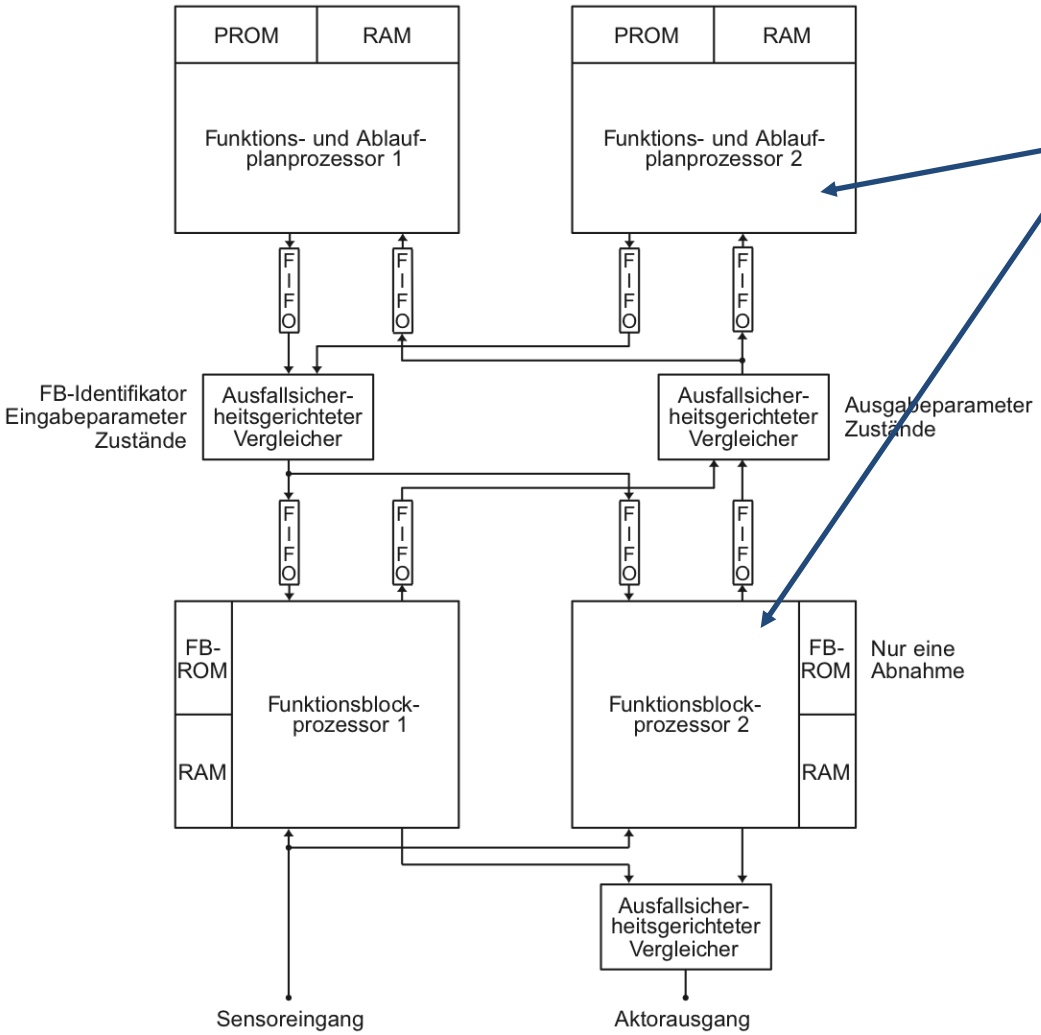
[1] W. A. Halang und R. M. Konakovsky, 2013

Das Funktionsplanparadigma bildet die Basis zur Realisierung einer Datenverarbeitungsanlage für sicherheitsgerichtete Automatisierungsaufgaben



[2] W. A. Halang und M. Sniezek, Patent, 1998

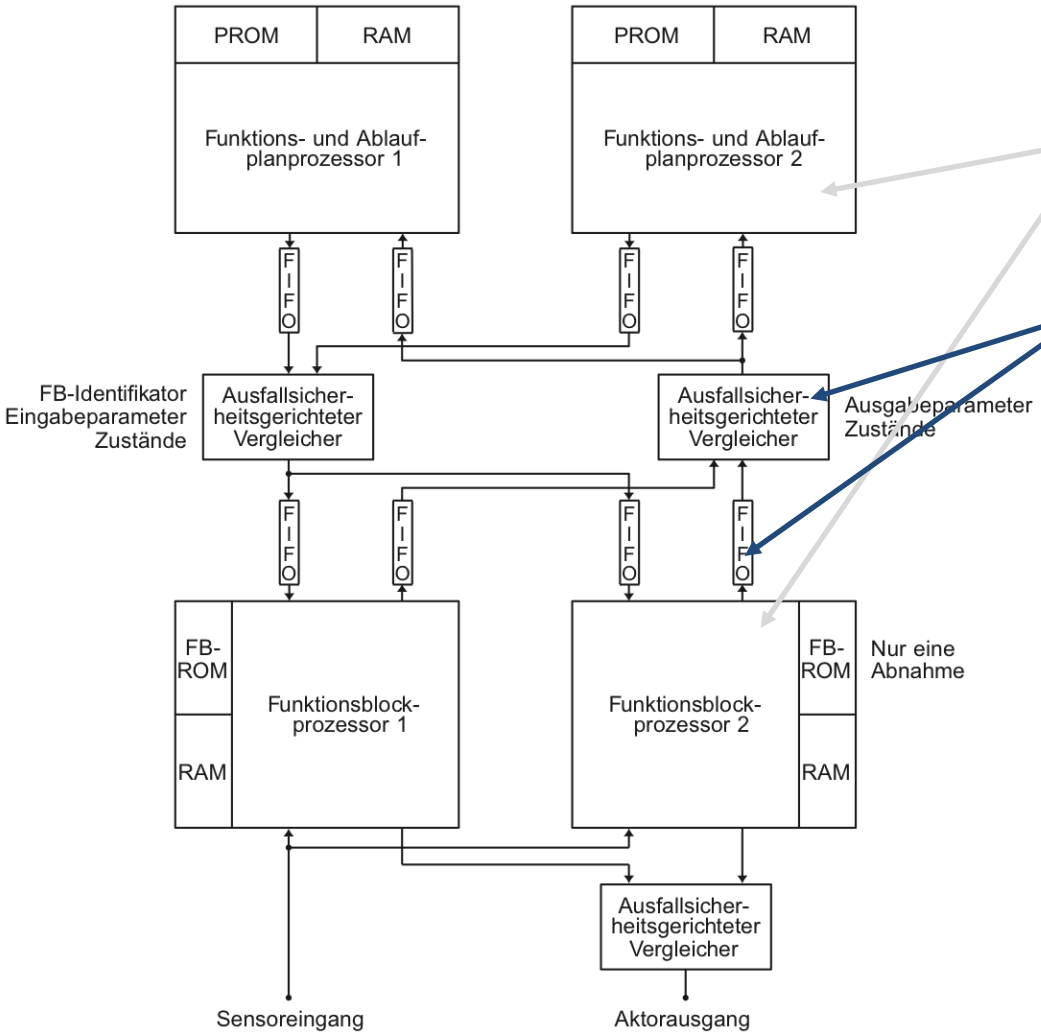
Die Datenverarbeitungsanlage für sicherheitsgerichtete Automatisierungsaufgaben ermöglicht eine vollständige Systemverifikation



Verifikation

Prozessoren:
Gegenstand einer aktuellen Dissertation

Die Datenverarbeitungsanlage für sicherheitsgerichtete Automatisierungsaufgaben ermöglicht eine vollständige Systemverifikation

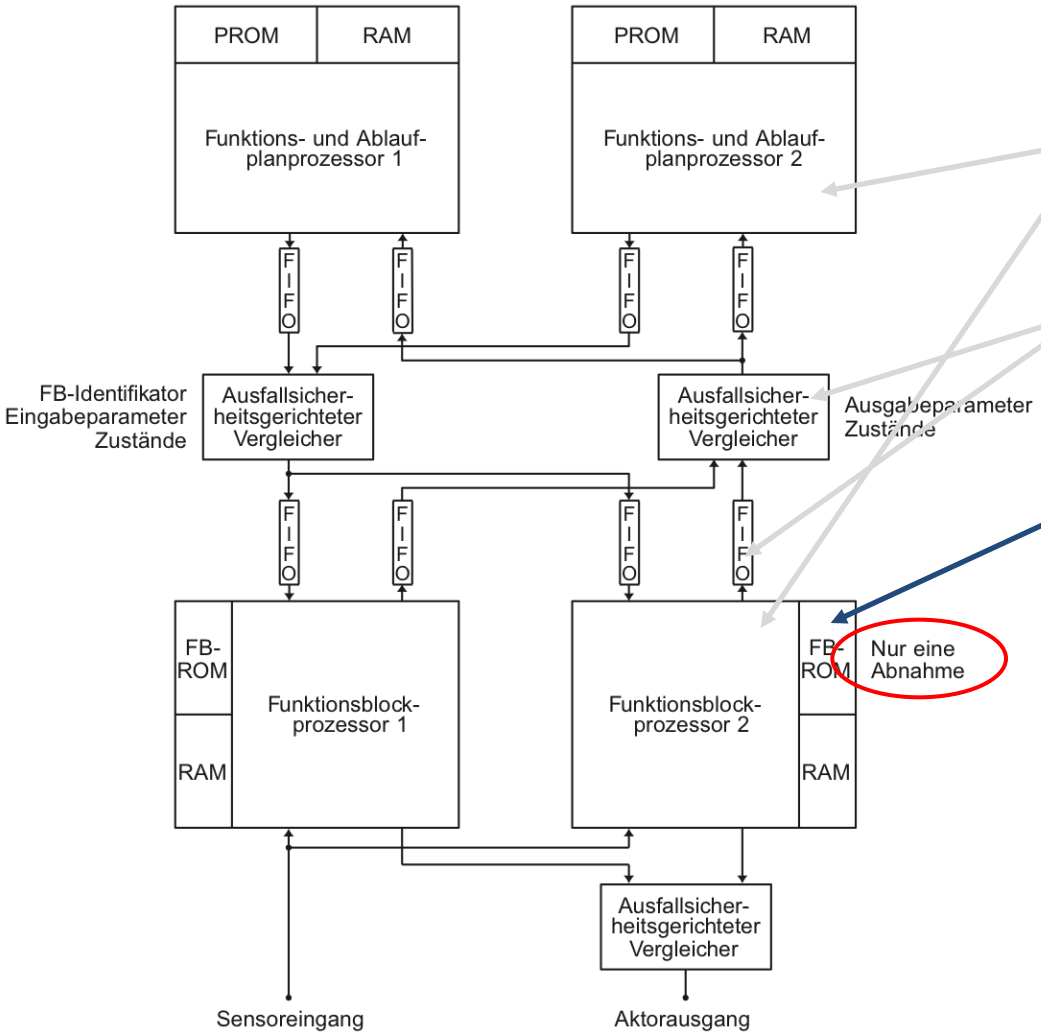


Verifikation

Prozessoren:
Gegenstand einer aktuellen Dissertation

FIFOs und Vergleicher:
TÜV-zertifizierte Logikgatter

Die Datenverarbeitungsanlage für sicherheitsgerichtete Automatisierungsaufgaben ermöglicht eine vollständige Systemverifikation



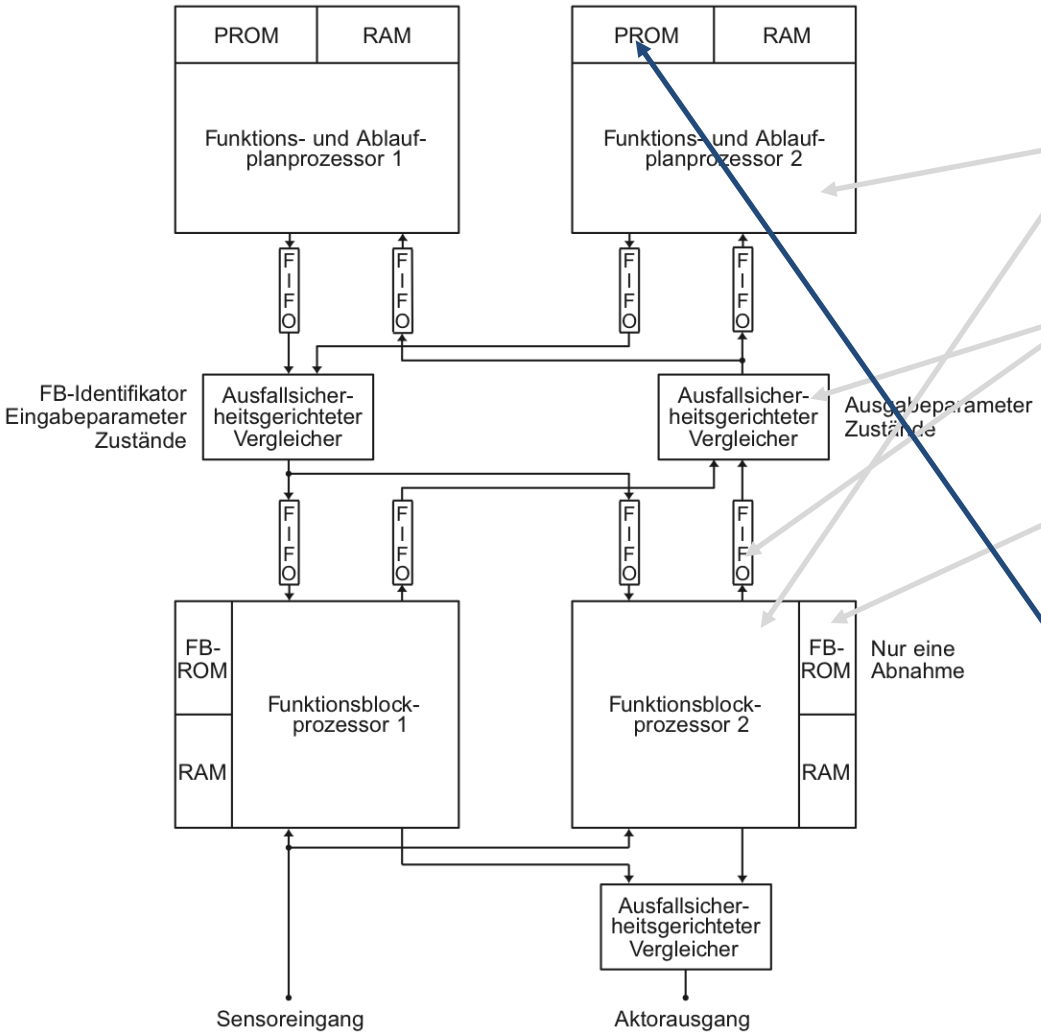
Verifikation

Prozessoren:
Gegenstand einer aktuellen Dissertation

FIFOs und Vergleicher:
TÜV-zertifizierte Logikgatter

Funktionsblockbibliothek:
Gegenstand dieser Arbeit
und diversitäre Rückübersetzung

Die Datenverarbeitungsanlage für sicherheitsgerichtete Automatisierungsaufgaben ermöglicht eine vollständige Systemverifikation



Verifikation

Prozessoren:
Gegenstand einer aktuellen Dissertation

FIFOs und Vergleicher:
TÜV-zertifizierte Logikgatter

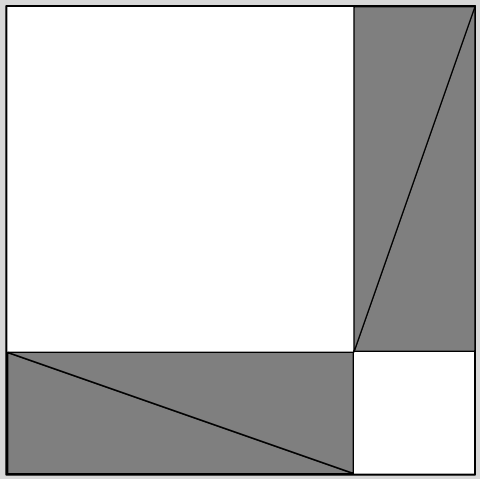
Funktionsblockbibliothek:
Gegenstand dieser Arbeit
und diversitäre Rückübersetzung

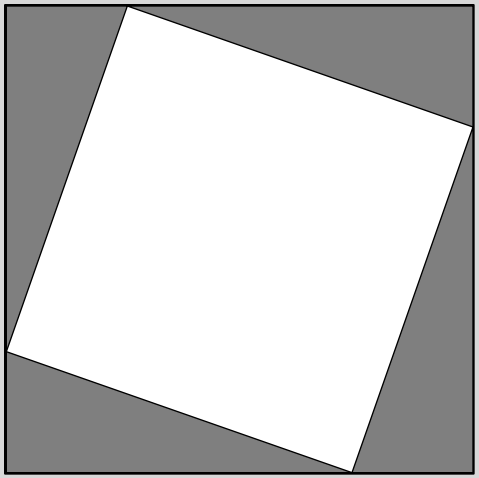
Funktions- und Ablaufpläne:
Diversitäre Rückübersetzung

Beweis des Satzes
durch Anwendung
mathematischer Methoden

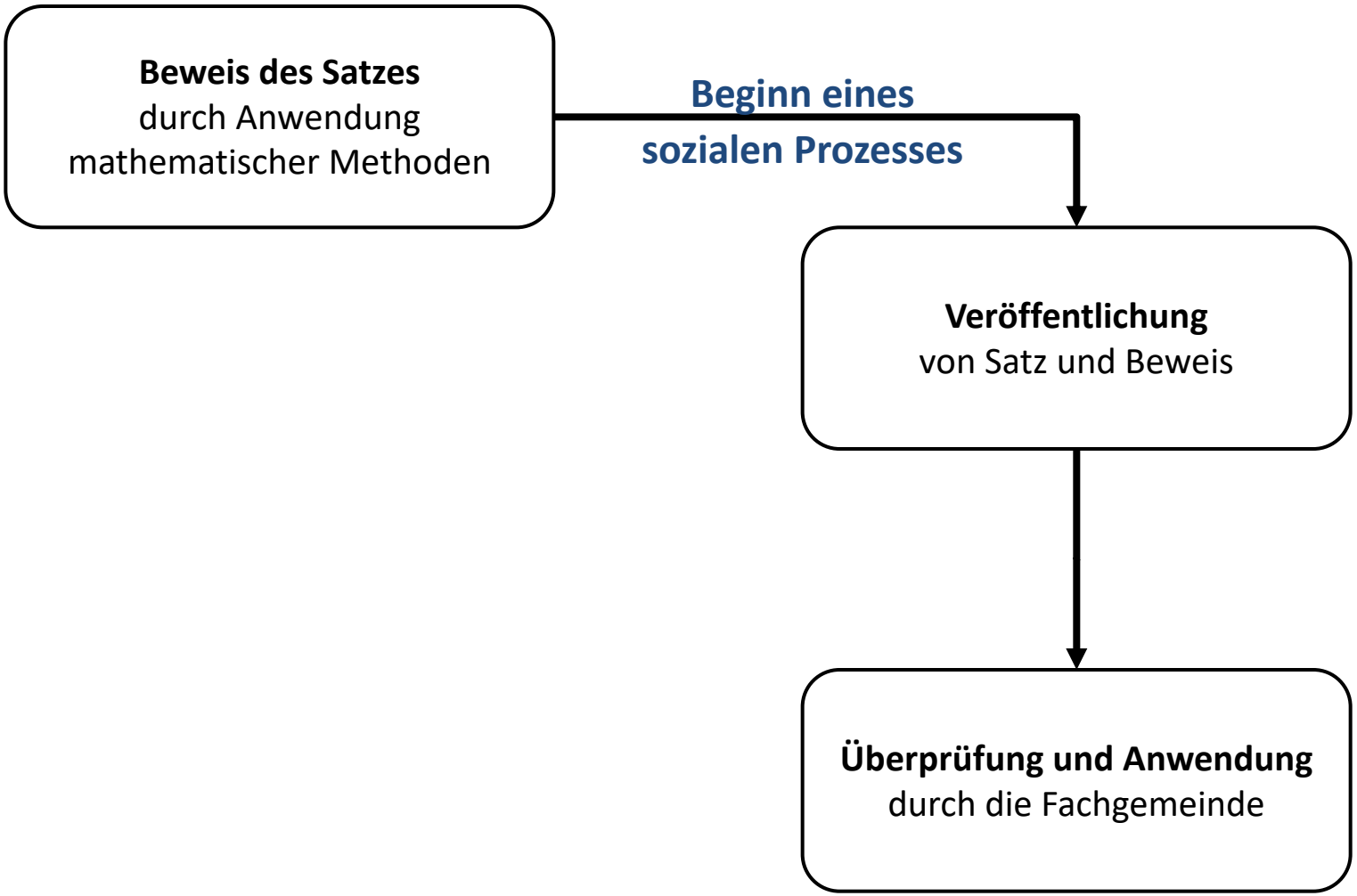
Beweis des Satzes
durch Anwendung
mathematischer Methoden

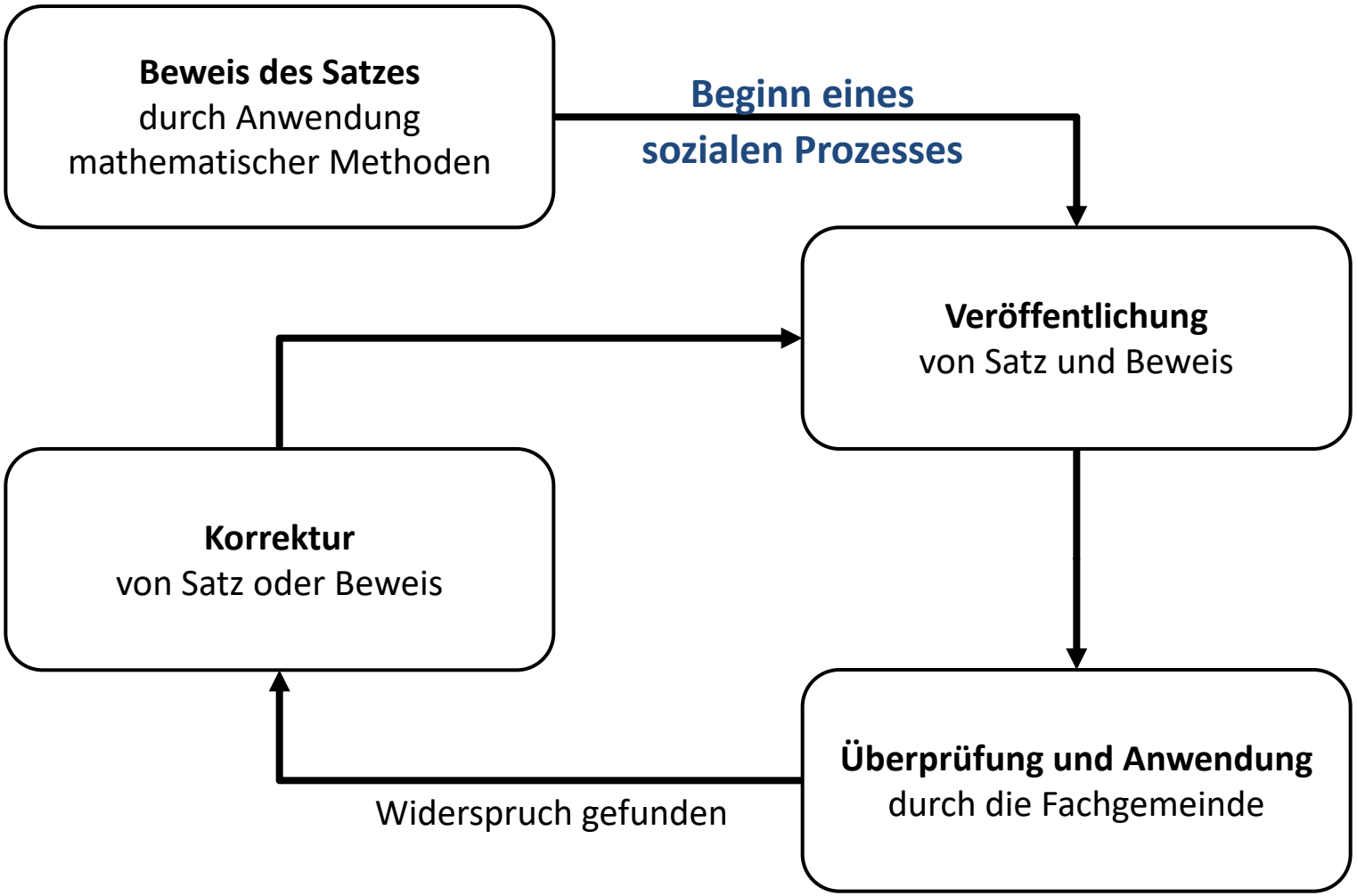
Beweis des Satzes des Pythagoras[4]





[4] R. Isaacs, 1975


Beweis des Satzes
durch Anwendung
mathematischer Methoden
Beginn eines
sozialen Prozesses
Veröffentlichung
von Satz und Beweis
Überprüfung und Anwendung
durch die Fachgemeinde


Beweis des Satzes
durch Anwendung
mathematischer Methoden
Beginn eines
sozialen Prozesses
Veröffentlichung
von Satz und Beweis
Überprüfung und Anwendung
durch die Fachgemeinde
Widerspruch gefunden
Korrektur
von Satz oder Beweis

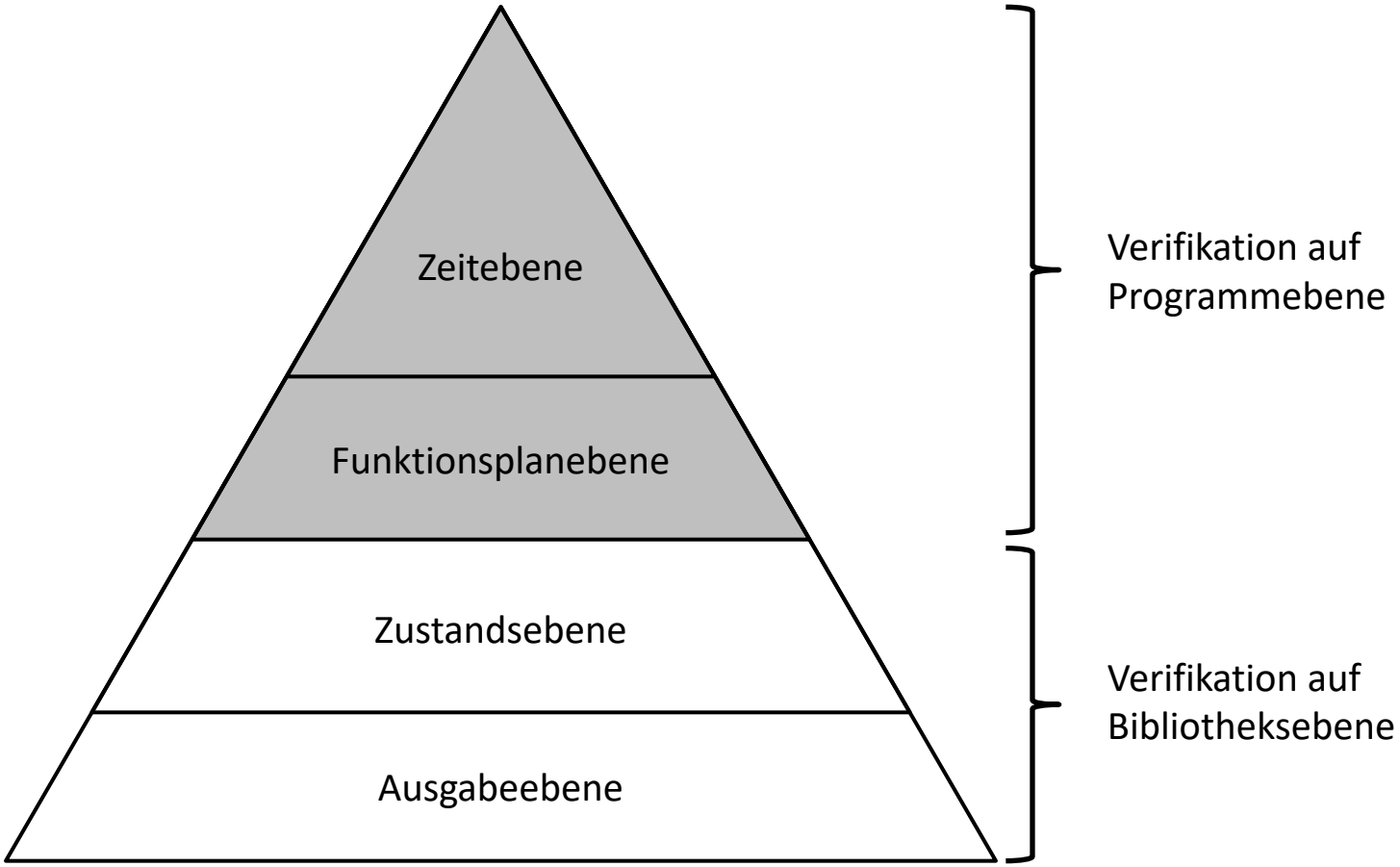
Durch Orientierung an der Richtlinie VDI/VDE 3696 ist die Bibliothek geeignet, die weitaus meisten Aufgaben der Prozessautomatisierung zu lösen

Funktionsblock	Verifikation
AND_	HW
NOT_	HW
OR_	HW
XOR_	HW
BITSHIFT	**QEB**
ADD_	**HW/QEB**
SUB_	**HW/QEB**
MUL_	**QEB**
DIV_	**QEB**
MOD_	**QEB**
BIT_N	**QEB**
MUX	**QEB**
DEMUX	**QEB**
MAX_	**QEB**
MIN_	**QEB**
LT_	**QEB**
GT_	**QEB**
LE_	**QEB**
GE_	**QEB**
EQ_	**QEB**
NE_	**QEB**
LIMIT_	**QEB**
SORT	**QEB**

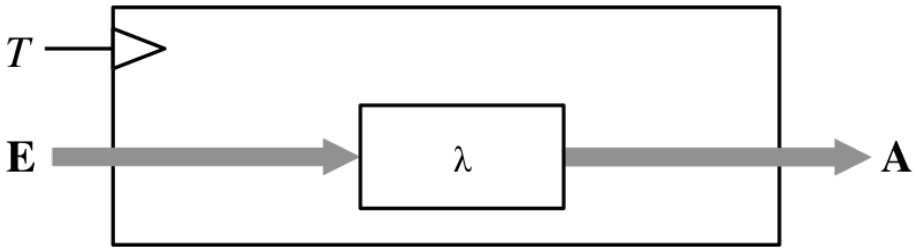
Funktionsblock	Verifikation
ABS_	**QEB**
SCAL_	**QEB**
NONLIN_	**QEB**
ACOS_	APPR
ASIN_	APPR
ATAN_	APPR
COS_	APPR
SIN_	APPR
TAN_	APPR
SQRT_	APPR
LN_	APPR
LOG_	APPR
EXP_	APPR
EXPT_	**QEB**
IN_A	HW
IN_B	HW
OUT_A	HW
OUT_B	HW
IN_CB	HW
IN_CN	HW
OUT_CB	HW
OUT_CN	HW
IN_W	HW

Funktionsblock	Verifikation
OUT_W	HW
RTRIG	**QEB (Z)**
FTRIG	**QEB (Z)**
RSFF	**QEB (Z)**
SRFF	**QEB (Z)**
TP1	**QEB (Z)**
TON1	**QEB (Z)**
TOF1	**QEB (Z)**
CT	**QEB (Z)**
CTD	**QEB (Z)**
PWM	**QEB (Z)**
R	**QEB (Z)**
REDUCT_AVG	**QEB (Z)**
REDUCT_MAX	**QEB (Z)**
REDUCT_MIN	**QEB (Z)**
SHYS	**QEB (Z)**
RATE_LIMIT	**QEB (Z)**
DIF	**QEB (Z)**
FIO	**QEB (Z)**
SEO	**QEB (Z)**
INTEGR	**QEB (Z)**
DEADT	**QEB (Z)**
AVER	**QEB (Z)**

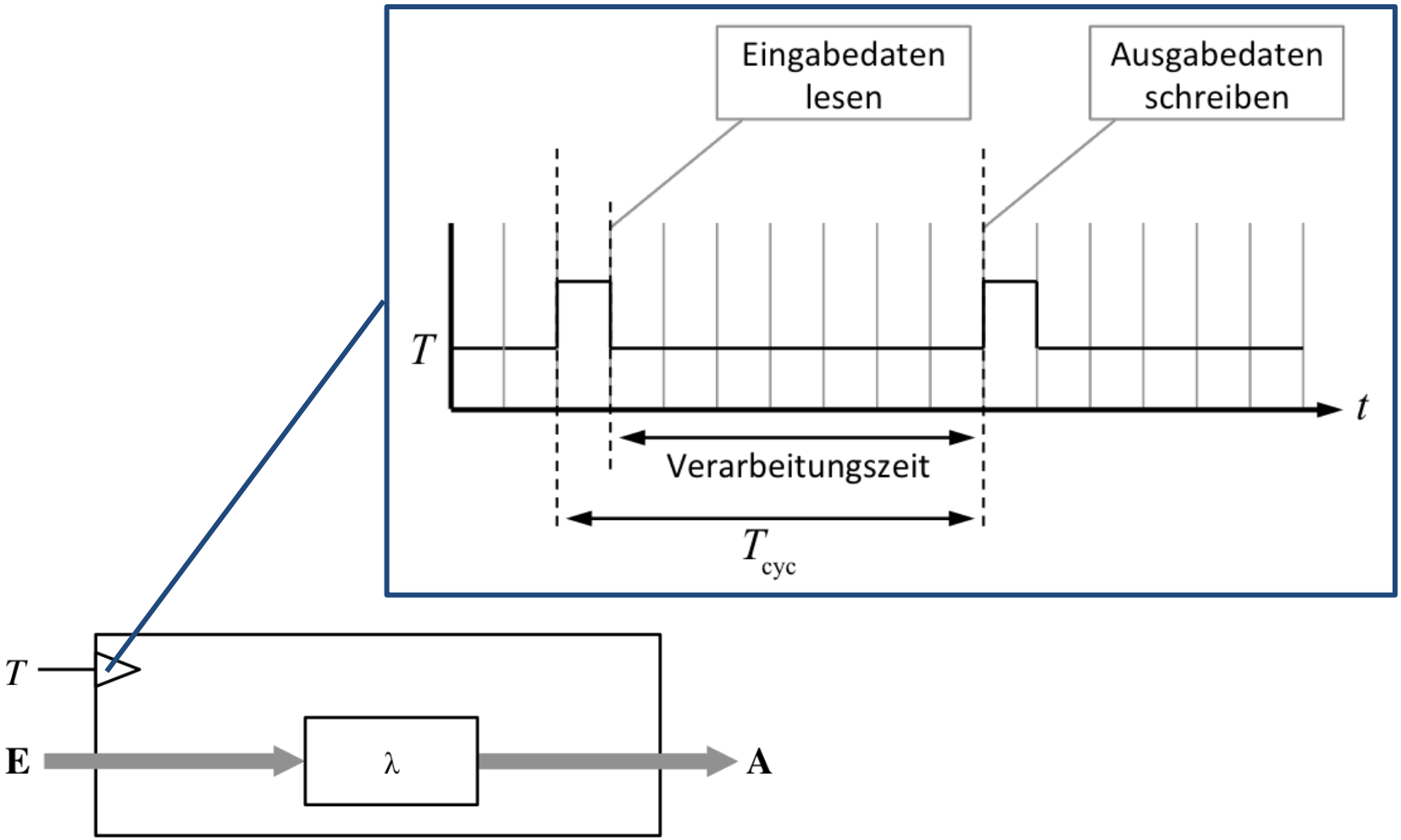
Zerlegen eines Programms in Verifikationsebenen macht die Komplexität beherrschbar



Funktionsbaustein OHNE interne Zustände

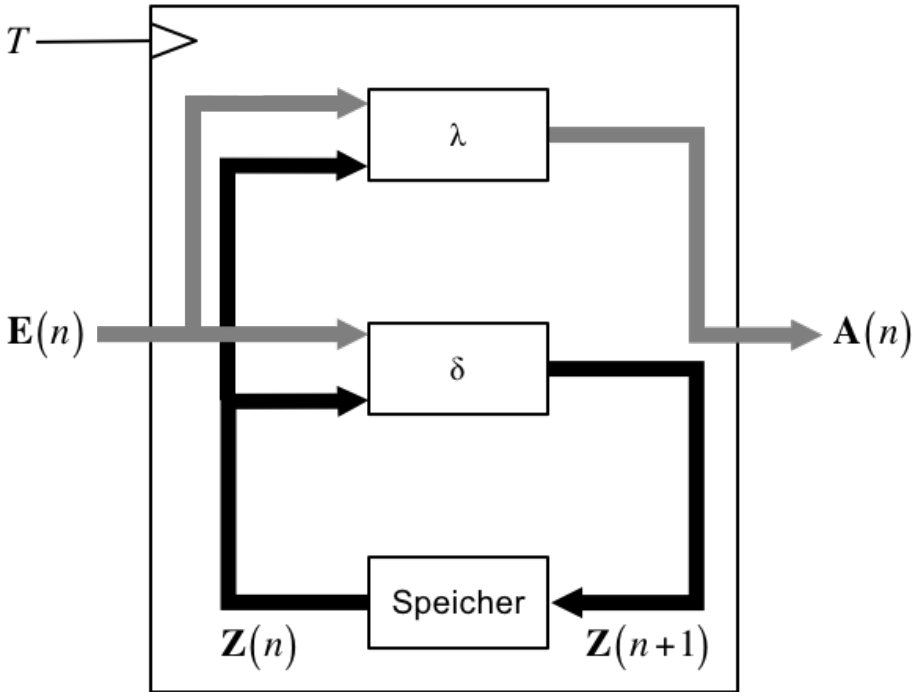


Korrektheit der Ausgabefunktion λ ist zu beweisen



[5] A. Benveniste und G. Berry, 1991

Funktionsbaustein MIT internen Zuständen



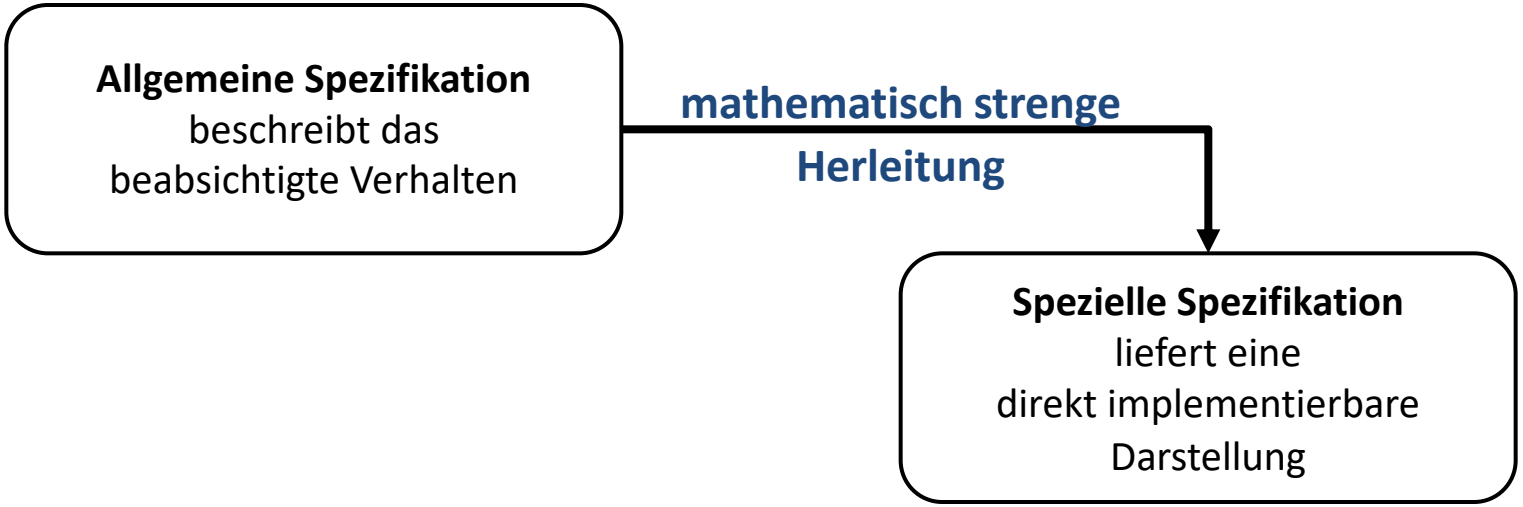
Korrektheit der Ausgabefunktion λ ist zu beweisen

Korrektheit der Überführungsfunktion δ ist zu beweisen

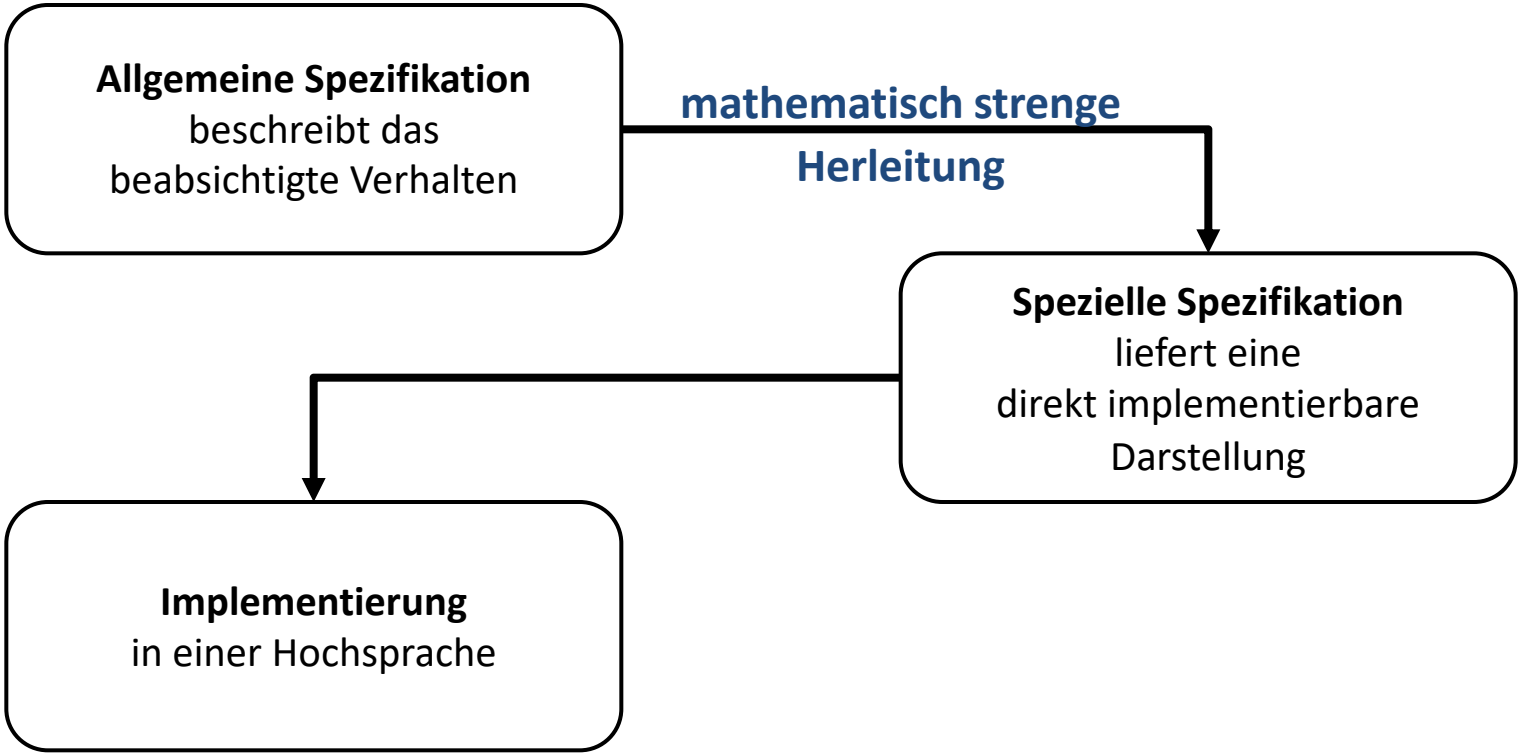
Durch entwurfsbegleitende Verifikation ergeben sich strukturell einfache Implementierungen und Beweise

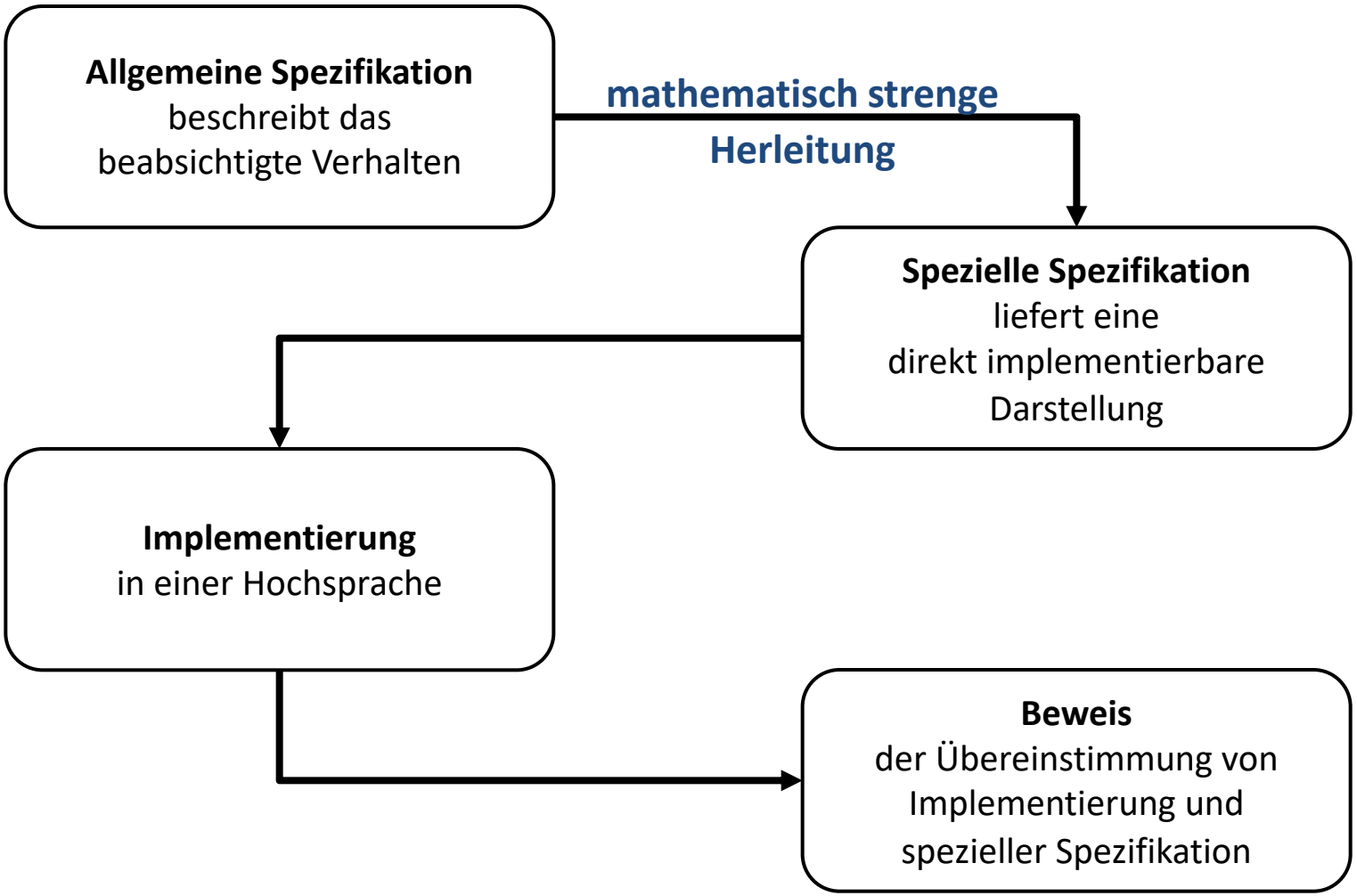
Allgemeine Spezifikation
beschreibt das
beabsichtigte Verhalten

Durch entwurfsbegleitende Verifikation ergeben sich strukturell einfache Implementierungen und Beweise



Durch entwurfsbegleitende Verifikation ergeben sich strukturell einfache Implementierungen und Beweise




Allgemeine Spezifikation
beschreibt das
beabsichtigte Verhalten
mathematisch strenge
Herleitung
Spezielle Spezifikation
liefert eine
direkt implementierbare
Darstellung
Implementierung
in einer Hochsprache
Beweis
der Übereinstimmung von
Implementierung und
spezieller Spezifikation

Allgemeine Spezifikation
beschreibt das
beabsichtigte Verhalten

DIF Differentiation (D-T1-Glied)

Der Funktionsbaustein soll das durch die Differentialgleichung

$$T_1 \cdot \dot{V}(t) + V(t) = T_D \cdot \dot{U}(t)$$

beschriebene Verhalten eines D-T1-Glieds annähern mit

$U(t)$ als Eingangsgröße,
$V(t)$ als Ausgangsgröße,
T_D als Differenzierbeiwert,
T_1 als Zeitkonstante.

Die Zeitkonstante T_1 soll dem Funktionsbaustein dabei nicht direkt sondern über eine Eingangsgröße

$$\mathrm{T1TOTD} = T_1/T_D$$

übergeben werden.

Allgemeine Spezifikation
beschreibt das
beabsichtigte Verhalten

Spezielle Spezifikation
liefert eine
direkt implementierbare
Darstellung

Differentialgleichung:

$$T_1 \cdot \dot{V}(t) + V(t) = T_D \cdot \dot{U}(t)$$

Numerische Näherung der zeitlichen Ableitung nach (Knorrenschild, 2013):

$$\dot{U}(n) = \frac{U(n) - U(n-1)}{T_{cyc}}$$

Durch Einsetzen und Umstellen nach $V(n)$ folgt

$$T_1 \cdot \frac{V(n) - V(n-1)}{T_{cyc}} + V(n) = T_D \cdot \frac{U(n) - U(n-1)}{T_{cyc}}$$

$$\Leftrightarrow \left(\frac{T_1}{T_{cyc}} + 1\right) V(n) - \frac{T_1}{T_{cyc}} V(n-1) = \frac{T_D}{T_{cyc}} \left(U(n) - U(n-1)\right)$$

$$\Leftrightarrow V(n) = \frac{\frac{T_1}{T_{cyc}} V(n-1) + \frac{T_D}{T_{cyc}} \left(U(n) - U(n-1)\right)}{\left(\frac{T_1}{T_{cyc}} + 1\right)}$$

und damit

$$T_1 = T1TOTD \cdot T_D$$

$$\alpha = T_1 / T_{cyc}$$

$$V(n) = \frac{\alpha \cdot V(n-1) + \frac{T_D}{T_{cyc}} \left(U(n) - U(n-1)\right)}{(1 + \alpha)}$$

Allgemeine Spezifikation
beschreibt das
beabsichtigte Verhalten

Spezielle Spezifikation
liefert eine
direkt implementierbare
Darstellung

Eingaben $E(n)$: $U(n), T_{cyc}, T_D, T1TOTD$

Ausgaben $A(n)$: $V(n)$

Zustand $Z(n)$:

$$U_{OLD}(n) = U(n-1)$$
$$V_{OLD}(n) = V(n-1)$$

Anfangszustand $Z(n \leq 0)$:

$$U_{OLD}(n \leq 0) = 0$$
$$V_{OLD}(n \leq 0) = 0$$

Ausgabefunktion λ:

$$T_1 = T1TOTD \cdot T_D$$
$$\alpha = T_1 / T_{cyc}$$
$$V(n) = \frac{\alpha \cdot V_{OLD}(n) + \frac{T_D}{T_{cyc}}\left(U(n) - U_{OLD}(n)\right)}{(1+\alpha)}$$

Überführungsfunktion δ:

$$U_{OLD}(n+1) = U(n)$$
$$V_{OLD}(n+1) = V(n)$$

Allgemeine Spezifikation
beschreibt das
beabsichtigte Verhalten

Spezielle Spezifikation
liefert eine
direkt implementierbare
Darstellung

Implementierung
in einer Hochsprache

```
FUNCTION_BLOCK DIF
VAR_INPUT
    U       : REAL;
    TCYC    : REAL;
    TD      : REAL := 1.0;
    T1TOTD  : REAL := 0.1;
END_VAR
VAR_OUTPUT
    V : REAL;
END_VAR
VAR_TEMP
    T1, ALPHA : REAL;
END_VAR
VAR
    U_OLD : REAL := 0.0;
    V_OLD : REAL := 0.0;
END_VAR

T1    := T1TOTD*TD;
ALPHA := T1/TCYC;
V     := (ALPHA*V_OLD + TD/TCYC*(U-U_OLD))
         / (1.0 + ALPHA);

U_OLD := U;
V_OLD := V;

END_FUNCTION_BLOCK
```

$E(n)$ $A(n)$ $Z(n)$ λ δ

Allgemeine Spezifikation
beschreibt das
beabsichtigte Verhalten

Spezielle Spezifikation
liefert eine
direkt implementierbare
Darstellung

Implementierung
in einer Hochsprache

Beweis
der Übereinstimmung von
Implementierung und
spezieller Spezifikation

Anfangszustand $Z(n \leq 0)$:

$$U_{OLD}(n \leq 0) = 0$$
$$V_{OLD}(n \leq 0) = 0$$

Ausgabefunktion λ:

$$T_1 = T1TOTD \cdot T_D$$
$$\alpha = T_1 / T_{cyc}$$
$$V(n) = \frac{\alpha \cdot V_{OLD}(n) + \frac{T_D}{T_{cyc}}\left(U(n) - U_{OLD}(n)\right)}{(1+\alpha)}$$

Überführungsfunktion δ:

$$U_{OLD}(n+1) = U(n)$$
$$V_{OLD}(n+1) = V(n)$$

```
U_OLD : REAL := 0.0;
V_OLD : REAL := 0.0;
```
$Z(n \leq 0)$

```
T1    := T1TOTD*TD;
ALPHA := T1/TCYC;
V     := (ALPHA*V_OLD + TD/TCYC*(U-U_OLD))
         / (1.0 + ALPHA);
```
λ

```
U_OLD := U;
V_OLD := V;
```
δ

Allgemeine Spezifikation
beschreibt das
beabsichtigte Verhalten

Spezielle Spezifikation
liefert eine
direkt implementierbare
Darstellung

Implementierung
in einer Hochsprache

Beweis
der Übereinstimmung von
Implementierung und
spezieller Spezifikation

Anfangszustand $Z(n \leq 0)$:

$$U_{OLD}(n \leq 0) = 0$$
$$V_{OLD}(n \leq 0) = 0$$

Ausgabefunktion λ:

$$T_1 = T1TOTD \cdot T_D$$
$$\alpha = T_1 / T_{cyc}$$
$$V(n) = \frac{\alpha \cdot V_{OLD}(n) + \frac{T_D}{T_{cyc}}\left(U(n) - U_{OLD}(n)\right)}{(1+\alpha)}$$

Überführungsfunktion δ:

$$U_{OLD}(n+1) = U(n)$$
$$V_{OLD}(n+1) = V(n)$$

```
U_OLD : REAL := 0.0;
V_OLD : REAL := 0.0;
```
$Z(n \leq 0)$

```
T1    := T1TOTD*TD;
ALPHA := T1/TCYC;
V     := (ALPHA*V_OLD + TD/TCYC*(U-U_OLD))
         / (1.0 + ALPHA);
```
λ

```
U_OLD := U;
V_OLD := V;
```
δ

Allgemeine Spezifikation
beschreibt das
beabsichtigte Verhalten

Spezielle Spezifikation
liefert eine
direkt implementierbare
Darstellung

Implementierung
in einer Hochsprache

Beweis
der Übereinstimmung von
Implementierung und
spezieller Spezifikation

Anfangszustand $Z(n \leq 0)$:

$$U_{OLD}(n \leq 0) = 0$$
$$V_{OLD}(n \leq 0) = 0$$

Ausgabefunktion λ:

$$T_1 = T1TOTD \cdot T_D$$
$$\alpha = T_1 / T_{cyc}$$
$$V(n) = \frac{\alpha \cdot V_{OLD}(n) + \frac{T_D}{T_{cyc}}\left(U(n) - U_{OLD}(n)\right)}{(1+\alpha)}$$

Überführungsfunktion δ:

$$U_{OLD}(n+1) = U(n)$$
$$V_{OLD}(n+1) = V(n)$$

```
U_OLD : REAL := 0.0;
V_OLD : REAL := 0.0;
```
$Z(n \leq 0)$

```
T1    := T1TOTD*TD;
ALPHA := T1/TCYC;
V     := (ALPHA*V_OLD + TD/TCYC*(U-U_OLD))
         / (1.0 + ALPHA);
```
λ

```
U_OLD := U;
V_OLD := V;
```
δ

Identifikation des quasiempirischen Beweises
als geeignete Verifikationsstrategie

Verlässliche Software für sicherheitsgerichtete Automatisierungssysteme

Identifikation des quasiempirischen Beweises
als geeignete Verifikationsstrategie

Entwicklung eines Modells zur Aufteilung in
unabhängige Verifikationsebenen

Verlässliche Software für sicherheitsgerichtete Automatisierungssysteme

- Identifikation des quasiempirischen Beweises als geeignete Verifikationsstrategie
- Entwicklung eines Modells zur Aufteilung in unabhängige Verifikationsebenen
- Verifikation einer vollständigen Bibliothek bestehend aus **44 Funktionsblöcken**

Verlässliche Software für sicherheitsgerichtete Automatisierungssysteme

- Identifikation des quasiempirischen Beweises als geeignete Verifikationsstrategie
- Entwicklung eines Modells zur Aufteilung in unabhängige Verifikationsebenen
- Verifikation einer vollständigen Bibliothek bestehend aus **44 Funktionsblöcken**
- Nachweis der Anwendbarkeit der diversitären Rückübersetzung

Verlässliche Software für sicherheitsgerichtete Automatisierungssysteme