Transparenter Steuerungsentwurf mit SFC nach IEC 1131-3

Dipl.-Ing. Georg Frey, Prof. Dr.-Ing. habil. Lothar Litz

Lehrstuhl für Automatisierungstechnik, Fachbereich Elektrotechnik
Universität Kaiserslautern, Postfach 3049, 67653 Kaiserslautern
Tel. 0631 - 205 - 4208, Fax. -4462, e-mail: frey@e-technik.uni-kl.de,
WWW: http://www.e-technik.uni-kl.de/litz

Im vorliegenden Beitrag werden Methoden zum transparenten Steuerungsentwurf mit SFC vorgestellt. Aus einer allgemeinen Betrachtung des Transparenzbegriffes werden konkrete Forderungen an den SFC-Entwurf entwickelt.

Transparent Controler Design with SFC following IEC 1131-3

In this paper we present methods for transparent controler design in the SFC-context. Starting at a proper definition of transparency some rules for transparent SFC design are introduced.

1 Einleitung

Mit der internationalen Norm IEC 1131-3 (bzw. DIN-IEC 61131-3) für die Erstellung von Programmen speicherprogrammierbarer Steuerungen (SPS) setzt sich zur Zeit ein neuer Programmierstandard durch. Kennzeichnend für die Entwicklung der Steuerungstechnik ist, daß Hardware-orientiertes Denken die Software-Lösungen beeinflußt hat [Litz97]. Dies ist auch der IEC 1131-3 anzumerken, die z.B. neben dem Strukturierten Text (ST = Structered Text) und dem SFC (Sequential Function Chart, deutsch AS, Ablaufsprache),(SFC = Sequential Function Chart) auch die Hardware-orientierten Darstellungsmittel Funktionplan (FBD = Function Block Diagram) und Kontaktplan (LD = Ladder Diagram) unterstützt. Normen wie die erwähnte IEC 1131-3 legen Realisierungsarten fest. Sie sorgen für die Unabhängigkeit der Programme von einer Hardware-Plattform. Sie garantieren aber keineswegs einen sauberen Steuerungsentwurf, da sie nicht zur Nutzung bestimmter Methoden zwingen. Sogar beliebig intransparente und kaum überprüfbare Steuerungsprogramme können - z.B. auf SFC-Basis - normenkonform erzeugt werden.

Neben praktischen Aspekten, wie der Möglichkeit, Steuerungsalgorithmen herstellerunabhängig zu entwerfen, bietet die Norm auch verbesserte Voraussetzungen zum strukturierten Steuerungsentwurf. Speziell der SFC, erlaubt es auch komplexe Steuerungsaufgaben strukturiert zu lösen. Zudem bietet der SFC, durch die grafische Darstellung der Steuerungsalgorithmen, auch eine gute Basis für eine Dokumentation der entwickelten Programme.

Tagungsband SPS/IPC/DRIVES'97 Nürnberg, S. 240-249, Hüthig Verlag.

In diesem Beitrag wird gezeigt, wie SFC-Programme transparent entworfen werden können. Zunächst wird der Begriff der Transparenz definiert. Darauf aufbauend wird gezeigt, wie man Transparenz beim Steuerungsentwurf grundsätzlich erreichen kann. Nach diesen Vorarbeiten werden verschiedene Funktionen des SFC auf Transparenz untersucht. Bei dieser Untersuchung ergeben sich einige Folgerungen, die anschließend genutzt werden um verschiedene Methoden zum transparenten SFC-Entwurf aufzuzeigen. Ein abschließender Vergleich der vorgestellten Methoden gibt Hinweise auf die möglichen Einsatzgebiete.

2 Der Transparenzbegriff

Transparenz bedeutet nach Litz [Litz95], daß eine entworfene Steuerung in bezug auf ihre Ein- und Ausgänge und ihren internen Zustand möglichst einfach und eindeutig erkennen läßt,

- was sie im Moment tun soll,
- was sie im Moment tatsächlich tut und
- was sie in nächster Zukunft tun soll.

Die Einhaltung dieser Kriterien bewirkt auch, daß eine transparent entworfene Steuerung rückinterpretierbar ist d.h. daß die Steuerungsaufgabe aus der Steuerung abgelesen werden kann. Transparenz erhöht somit neben der Funktionalität einer Steuerung auch ihre Verständlichkeit. Diese Folgerung ist immens wichtig, da bei industriellen Steuerungen folgende Gegebenheiten angetroffen werden [Litz95]:

- eine hohe Anzahl zu verarbeitender Signale,
- Know-How-Geber und Planer sind nicht unbedingt Automatisierungstechniker,
- wechselnde Bearbeiter beim Entwurf und Betrieb und
- ein lebendiger Steuerungsalgorithmus, der im Laufe der Zeit Änderungen unterliegt.

Eine transparent entworfene Steuerung führt gerade unter diesen Bedingungen zu erheblicher Zeitersparnis bei humanintensiven und deshalb teuren Aufgaben wie Inbetriebnahme, Fehlersuche und Programmänderung. Außerdem wird die Einarbeitung eines neuen Programmierers erleichtert bzw. oft überhaupt erst ermöglicht.

3 Transparenter Steuerungsentwurf

Ob es gelingt, eine Steuerung transparent zu entwerfen, hängt zunächst entscheidend von der gewählten Entwurfsmethode bzw. dem verwendeten Werkzeug ab. Die Anweisungsliste (AWL), die auf der Registerdenkweise beruht, stellt beispielsweise keine Möglichkeit zur transparenten Abbildung bedingter Verzweigungen (IF, THEN,

ELSE) zur Verfügung. Bei der Realisation über bedingte Sprünge geht die kausale Struktur vollständig verloren und der Code ist kaum noch rückinterpretierbar.

Neuere Sprachen mit einem mächtigen Funktionsumfang wie SFC oder ST (Structured Text) hingegen, verleiten den Programmierer oft dazu, die gebotene Funktionalität auszunutzen. Die durch „trickreiche“ Programmierung entstehenden intransparenten Algorithmen werden auch heute noch häufig mit der Effizienz des Programmes in bezug auf Codeumfang oder Anzahl der verwendeten Variablen gerechtfertigt. Bei ständig fallenden Hardwarekosten stehen derartige „Einsparungen“ jedoch in keinem Verhältnis zu den Kosten der Ressource Mensch, die bei Inbetriebnahme, Wartung und Änderung „optimaler“ Programme anfallen.

Transparenz kann folglich nur mit einer Methode erreicht werden, welche die nötigen Konstrukte zur transparenten Darstellung bietet. Dies sind vor allem:

- explizite Darstellung von Nebenläufigkeiten,
- explizite Darstellung kausaler Abhängigkeiten,
- explizite Darstellung zeitlicher Abhängigkeiten und
- klare Trennung zwischen Eingabe-, Ausgabe- und Zustandsvariablen.

Die Trennung der verschiedenen Variablenbereiche führt zu mehr Transparenz, weil der Informationsfluß deutlicher wird. Im Programm sollte eindeutig zwischen Ursache und Wirkung unterschieden werden können d.h. es muß klar erkennbar sein wo Informationen von gesteuerten Prozeß eingekoppelt werden und wo eine Informationsausgabe stattfindet. Bei ablauforientierten Programmiermethoden wird man folglich die Ausgaben den Zuständen des Algorithmus zuordnen und die Eingaben als Bedingungen für Zustandsübergänge nutzen.

Die Bereitstellung dieser Funktionalität ist jedoch nur eine notwendige Bedingung. Darüber hinaus sollten Konstrukte, die zur intransparenten Programmierung geeignet sind (wie z.B.: Sprungbefehle) erst gar nicht zur Verfügung stehen (hinreichende Bedingung).

4 Transparenzuntersuchung des SFC

Der SFC erfüllt die ersten drei der oben für eine transparente Programmentwicklung als notwendig angegebenen Bedingungen:

- Nebenläufigkeiten können in der grafischen Darstellung direkt und anschaulich wiedergegeben werden,
- kausale Abhängigkeiten sind durch die Verknüpfung an Transitionen ebenfalls in der grafischen Struktur leicht darstellbar und
- zeitliche Abhängigkeiten können durch Abfragen der Schrittmerkerzeiten an den Transitionen explizit berücksichtigt werden.

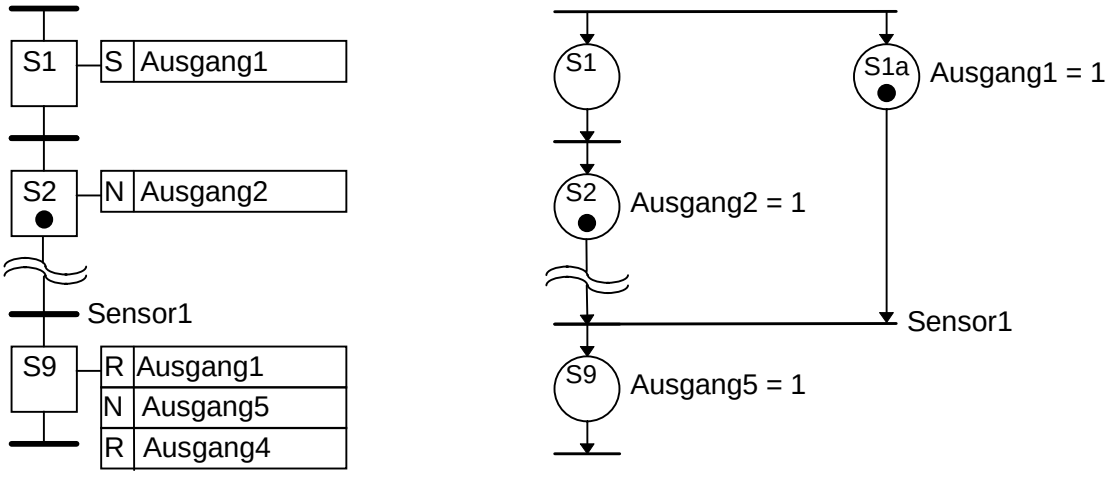
Die Trennung der Variablenbereiche ist im SFC nicht vorgesehen, kann aber durch den Entwickler erreicht werden, indem er auf die Verwendung von Ausgangssignalen in Transitionsbedingungen verzichtet und in den Schritten nicht auf Sensorsignale zugreift.

Darüber hinaus bietet der SFC jedoch noch weitere Funktionen, von denen einige im folgenden auf Ihre Transparenz untersucht werden sollen. Die Transparenzuntersuchung des SFC beruht auf dessen Umsetzung in funktionsgleiche interpretierte Petri-Netze (IPN) [KöQu88]. Im folgenden wird eine Überführung von SFC-Algorithmen verwendet, die über die rein strukturelle Übersetzung in [DaAl92] hinausgeht. Die Einzelheiten dieser Umsetzung sind in [Jörns96] beschrieben. Einen guten Überblick über die Syntax und Möglichkeiten zum Einsatz des SFC bietet z.B. [JoTi95], zur Theorie der Petri-Netze sowie der SIPN sei auf [Abel90] bzw. [KöQu88] verwiesen. Durch ihren überschaubaren Funktionsumfang und eine klar definierte Semantik sind IPN zur transparenten Darstellung von Steuerungsalgorithmen besonders geeignet. Die Umsetzung von SFC-Konstrukten in IPN zeigt, ob es sich um einen transparenten SFC handelt, da zum Beispiel verdeckte Nebenläufigkeiten des SFC im IPN direkt sichtbar werden.

Im einzelnen werden speichernde Befehle, die verschiedenen Möglichkeiten zur Darstellung zeitlicher Abhängigkeiten und die Einbindung beliebiger Programme in den SFC näher betrachtet [FrLi97].

4.1 Speichernde Befehle

Bild 1 zeigt einen kurzen SFC-Ablauf mit gespeichert ausgegebener Aktion (Bestimmungszeichen S) und ein funktionsgleiches IPN. Im SFC mit speicherndem Befehl (Bild 1a) kann anhand der Markierung nicht direkt auf die Ausgabe geschlossen werden. Man muß vielmehr das gesamte Netzwerk betrachten, um eventuell vorher speichernd gesetzte Ausgaben zu erkennen.



a) SFC mit speichernden Befehlen

b) funktionsgleiches IPN

Bild 1: Gespeicherte Ausgabe mit Bestimmungszeichen S und R

Das IPN kennt keine gespeicherten Aktionen. Deshalb kann man im funktionsgleichen IPN (Bild 1b) anhand der aktuellen Markierung sofort die Ausgaben ablesen. Die Überführung in ein IPN offenbart also versteckte Nebenläufigkeiten im SFC. Die (automatische) Umsetzung des IPN in einen funktionsgleichen SFC ohne speichernde Befehle (Bild 2) zeigt, wie auch im SFC transparent programmiert werden kann. Hieraus ergibt sich die erste Folgerung:

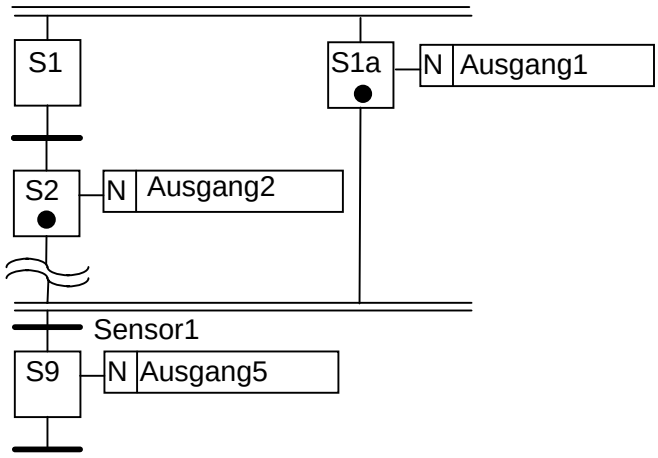


Bild 2: Transparenter SFC

Gespeichert ausgeführte Aktionen (Bestimmungszeichen S und R) erzeugen Intransparenz durch implizite Darstellung von Nebenläufigkeiten.

4.2 Zeitliche Abhängigkeiten

Die Behandlung zeitlicher Abhängigkeiten im SFC wird am Beispiel der Steuerungsaufgabe

„Starte Pumpe1 10 Sekunden nach Aktivierung des Überlaufsensors“

untersucht. Die Bilder 3 und 4 zeigen die Realisierung der Steuerungsaufgabe in zwei verschiedenen Darstellungsformen des SFC und die jeweils zugehörige Umsetzung in IPN.

Aktionen können im SFC über die Bestimmungszeichen D, L und P zeitgesteuert ausgeführt werden (Bild 3).

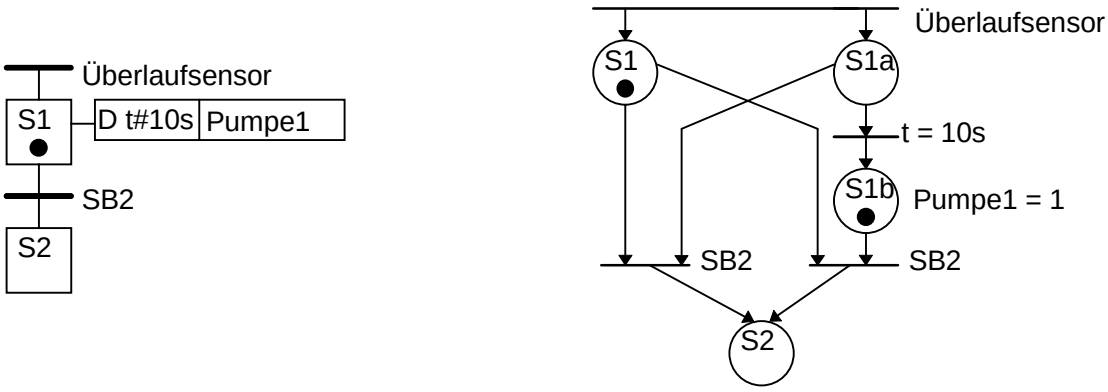


Bild 3: Verzögerte Ausgabe mit Bestimmungszeichen D.

Alternativ dazu kann die Zeitdauer, die seit der Aktivierung eines Schrittes verstrichen ist, im Netzwerk als Weiterschaltbedingung verwendet werden (Bild 4). Bei der Darstellung mit dem Bestimmungszeichen D ergibt sich ein komplexeres IPN als im zweiten Fall.

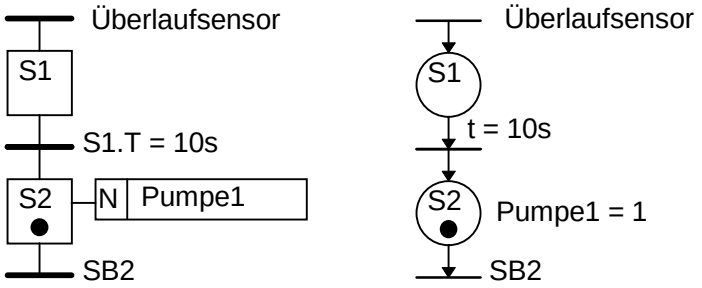


Bild 4: Verzögerte Ausgabe mit Schrittzeit.

Dies liegt daran, daß bei Schalten der Nachfolgetransition aus der Bearbeitung des Schrittes herausgesprungen wird. Im konkreten Fall hat das zur Folge, daß Pumpe1 nicht gestartet wird, wenn die Schaltbedingung SB2 vor Ablauf der 10 Sekunden erfüllt ist. Außerdem kann bei Verwendung der zeitverzögerten Ausgabe aus der Schrittmarkierung nicht mehr auf die momentane Ausgabe geschlossen werden. Bei Benutzung der Aktivierungszeiten als Schaltbedingung ergibt sich hingegen ein transparenter Ablauf. Dieser entspricht wohl meist auch mehr der Aufgabenstellung und gewährleistet somit die Rückinterpretierbarkeit der Steuerung. Für die Bestimmungszeichen L und P erhält man vergleichbare Ergebnisse. Daraus folgert man:

Die Bestimmungszeichen D, L und P erzeugen Intransparenz durch verdeckte temporale Abhängigkeiten.

4.3 Beliebige Programmeinbindung

Die Möglichkeit zur Hierarchiebildung, die durch Einbinden beliebiger Programme in einer der genormten Steuerungsfachsprachen in die Aktionen des SFC entsteht, kann einerseits zu deutlich besser strukturierten Programmen führen. Daß diese Hierarchiebildung andererseits nur dann transparent ist, wenn in den unteren Ebenen auch SFC verwendet wird, ist offensichtlich. Doch selbst dann können bei Zugriff auf globale Variablen - insbesondere auch fest adressierte Variablen - Informationsflüsse im SFC realisiert werden, die aus der Struktur nicht mehr ersichtlich sind. Deshalb ergeben sich zwei weitere Folgerungen:

Die Verwendung unterschiedlicher Programmiersprachen führt zu einem Bruch in der Darstellungsform und somit zu Intransparenz.

Die Verwendung globaler Variablen, kann verdeckte Informationsflüsse bewirken. D.h. kausale Abhängigkeiten werden nicht dargestellt und die Transparenz geht verloren.

4.4 Zusammenfassung

Zusammenfassend kann man sagen, daß der SFC prinzipiell den Entwurf transparenter Programme erlaubt. Durch die Bestimmungszeichen ist es jedoch möglich, Informationen stark zu verdichten. Informationsverdichtung führt aber leicht zu Intransparenz. Gerade in Verbindung mit der IEC 1131-3 erlaubt der SFC auch die intransparente Realisation versteckter Informationsflüsse. Deshalb gilt:

Ein SFC ohne Restriktionen ist in der Regel intransparent bis chaotisch.

5 Transparenter SFC-Entwurf

Hier gibt es drei Methoden, die jeweils einzeln oder auch in Kombination angewendet werden können. Wofür der Entwickler sich entscheidet, hängt von dessen persönlicher Präferenz und nicht zuletzt vom konkreten Anwendungsfall ab.

5.1 Synthese eines SFC aus dem IPN

Jedes IPN kann automatisch in einen SFC übersetzt werden. Die Synthese liefert einen transparenten Algorithmus, da nur ein Teil der Möglichkeiten des SFC genutzt wird, und die Struktur des IPN bei der Umsetzung erhalten bleibt. Der so gewonnene SFC enthält weder gespeicherte noch über die Bestimmungszeichen S, D, und L zeitgesteuert ausgegebene Aktionen. Es wird also mit impliziten Restriktionen gearbeitet.

Zur Analyse von IPN existieren formale Methoden, wie sie z.B. in dem Petri-Netz-basierten Steuerungsentwurfswerkzeug Netmate [FrLiJ97] realisiert sind. Deshalb kann auf der Basis des IPN, vor der Umsetzung in SFC eine Überprüfung des Algorithmus durchgeführt werden.

5.2 Analyse eines bestehenden SFC mittels IPN

Auch ein bereits entworfener SFC kann auf seine Transparenz analysiert werden. Die Umwandlung in ein funktionsgleiches IPN zeigt z.B. versteckte Nebenläufigkeiten und temporale Abhängigkeiten. Aus dem erzeugten IPN können Hinweise zur Verbesserung des zugrundeliegenden SFC gewonnen werden. Außerdem lassen sich so die Analysemethoden des IPN anwenden.

Diese Methode ist nur bei reinen SFC-Programmen anwendbar, da eine Umsetzung von eingebundenen AWL- oder KOP-Programmen im allgemeinen einen erheblichen Aufwand bedeutet.

5.3 Direkter SFC-Entwurf mit Restriktionen

Durch bewußte Restriktionen kann der Programmierer Transparenz direkt bei der SFC-Programmierung erzwingen. Einzelne Forderungen sind:

- explizite Darstellung von Nebenläufigkeiten durch Vermeidung der Bestimmungszeichen S und R,
- explizite Darstellung zeitlicher Abhängigkeiten durch Vermeidung der Bestimmungszeichen D, L und P,
- explizite Darstellung kausaler Abhängigkeiten durch möglichst wenige Zugriffe auf globale Variablen, sowie
- Trennung der Variablen, indem an den Transitionen keine Ausgangssignale eingekoppelt werden und in den Schritten keine Sensorsignale abgefragt werden.

Diese können z.B. in firmeninternen Programmierrichtlinien festgelegt werden. Besser wären allerdings Programmierwerkzeuge, in denen die Restriktionen bereits fest implementiert sind.

5.4 Vergleich der Entwurfsmethoden

Sofern der Entwickler sich nicht von der gewohnten Methode SFC trennen will, stellt der restringierte Entwurf nach 5.3 den direktesten und einfachsten Weg zum transparenten Algorithmus dar. Der Entwurf beliebiger Programme und deren nachträgliche Umsetzung (entsprechend 5.1) ist aufwendiger und daher nicht zu empfehlen. Diese Methode kann aber immer dann benutzt werden, wenn bereits bestehende Programmteile in ein neues Projekt einfließen, oder ein Reengineering vorhandener Software durchgeführt wird. Der Entwurf über IPN hat lediglich den Nachteil, daß eine Einarbeitung in die neue Methode erforderlich ist. Doch die Kenntnisse der Petri-Netze werden auch für das Verständnis formaler Überprüfungsergebnisse benötigt werden.

Formale Überprüfungen sind innerhalb des SFC derzeit nicht möglich. Nach der Umsetzung in ein funktionsgleiches IPN kann aber selbst ein intransparent entworfener SFC-Algorithmus formal überprüft werden [Jörns97]. Dabei können folgende Fragen beantwortet werden:

- Ist der Algorithmus deterministisch?
- Gibt es Endlosschleifen (ungewollte Zyklen)?
- Erfolgt zu jeder Zeit eine korrekte Ausgabe?
- Handelt es sich um ein sicheres SFC-Netzwerk?
- Gibt es im SFC-Netzwerk unerreichbare Ablaufketten?

Diese formale Überprüfbarkeit ist sowohl bei der Methode nach 5.1 als auch nach 5.2 direkt gegeben, kann aber auch beim restringierten SFC-Entwurf nach 5.3 durch nachträgliche Umsetzung in IPN erreicht werden.

Bild 5 zeigt nochmals die drei in den vorangegangenen Unterpunkten beschriebenen Entwurfsmethoden.

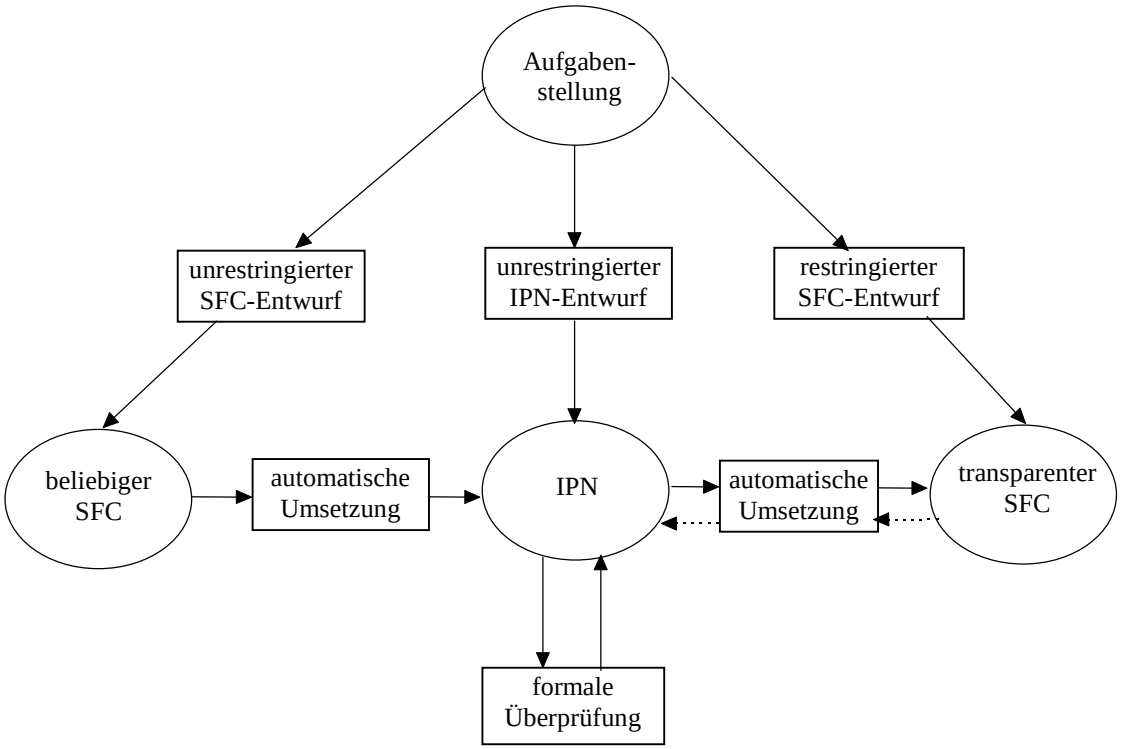


Bild 5: Entwurfsmethoden.

6 Zusammenfassung und Ausblick

In diesem Beitrag wurde gezeigt, daß der SFC gute Voraussetzungen für einen transparenten Steuerungsentwurf bietet. Leider erlaubt er aber auch völlig intransparenten Programmierstil. Durch die Untersuchung einzelner SFC-Konstrukte konnte festgestellt werden, wo die Transparenz eingeschränkt wird. SFC-Algorithmen, die ausschließlich N-Aktionen enthalten, erlauben einen guten Überblick über die erstellte Funktionalität. Durch den Einsatz speichernder und zeitgesteuerter Aktionen wird die eigentliche Komplexität des Steuerungsalgorithmus in den Action Control Block des SFC verlagert. Diese Verlagerung geht jedoch auf Kosten der Transparenz.

Viele Probleme beim Programmentwurf und vor allem bei der Wartung können vermieden werden, wenn der Entwickler sich an einige wenige Forderungen hält, die in diesem Beitrag aufgezeigt wurden. In jedem Fall ermöglicht die Überführung eines SFC-Algorithmus in ein IPN zum einen die Offenlegung der tatsächlichen Funktionalität. Zum anderen wird auf der Basis dieses IPN eine Analyse des SFC-

Steuerungsalgorithmus ermöglicht. Diese Analyse liefert als formale Überprüfung weitreichende Aussagen über den Steuerungsalgorithmus, die eine Inbetriebnahme und den Betrieb der Steuerung wesentlich erleichtern. In diesem Zusammenhang, wäre eine Integration der in den Hochschulen entwickelten Analysemethoden in kommerzielle Entwicklungstools wünschenswert.

Letztlich liegt es nun beim Anwender, was er aus den neuen Möglichkeiten mit den daraus resultierenden Einsparpotentialen bei Wartung, Änderung und Prüfung seiner Programme macht. Die Möglichkeit der formalen Überprüfung von Programmen wird in der Praxis bisher kaum genutzt. Hier bleibt abzuwarten, inwieweit z.B. erhöhte Aufwendungen bei der Zertifizierung sicherheitsrelevanter Programme zu einem Umdenken führen.

7 Literatur

[Abel90] Abel, D.: Petri-Netze für Ingenieure. Berlin: Springer-Verlag 1990.

[DaAl92] David, R.; Alla, H.: Petri nets and Grafcet - Tools for Modelling Discrete Event Systems. New York, London: Prentice Hall 1992.

[FrLi97] Frey, G., Litz, L.: Wie transparent sind SFC - Programme? In: Proc. GMA-Fachtagung Steuerungstechnik in Langen 10/97.

[FrLiJ97] Frey, G.; Litz, L.; Jörns, C.: Steuerungsentwurfstool Netmate. Automatisierungstechnik at 45 (1997) 7, S. 304-305, R. Oldenbourg Verlag.

[Jörns96] Jörns, C.; Litz, L.: Analyse und Verifikation von Steuerungsalgorithmen in der Ablaufsprache nach DIN EN 61131-3. In: Proc. SPS/IPC/DRIVES '96 Sindelfingen.

[Jörns97] Jörns, C.: Ein integriertes Steuerungsentwurfs- und Verifikationskonzept mit Hilfe interpretierter Petri-Netze. Dissertation, Universität Kaiserslautern 1996; VDI-Verlag Düsseldorf, Reihe8, Nr. 641.

[JoTi95] John, K.-H.; Tiegelkamp, M.: SPS-Programmierung mit IEC 1131-3. Berlin: Springer-Verlag 1995.

[KöQu88] König, R.; Quäck, L.: Petri-Netze in der Steuerungs- und Digitaltechnik. München, Wien: R. Oldenbourg Verlag 1988.

[Litz95] Litz, L.: Entwurf industrieller Prozeßsteuerungen auf der Basis geeigneter Petri-Netz-Interpretationen. In: Schnieder, E. (Hrsg.): Entwurf komplexer Automatisierungssysteme 1995. Braunschweig 1995, S. 417-429.

[Litz97] Litz, L.: Methoden und Werkzeuge zum Steuerungsentwurf - Historie, Stand, Ausblick. In: Proc. GMA-Fachtagung Steuerungstechnik in Langen 10/97.

[DIN IEC 61131-3] Programmable Controllers, Part 3: Programming languages.