Zeitschrift: Bulletin des Schweizerischen Elektrotechnischen Vereins, des Verbandes Schweizerischer Elektrizitätsunternehmen = Bulletin de l'Association suisse des électriciens, de l'Association des entreprises électriques suisses

Herausgeber: Schweizerischer Elektrotechnischer Verein ; Verband Schweizerischer Elektrizitätsunternehmen

Band: 86 (1995)

Heft: 21

Artikel: Spezifizieren und Entwickeln mit grafischen Modellen : CIP : eine konstruktive Entwicklungsmethode für eingebettete Systeme

Autor: Fierz, Hugo

DOI: https://doi.org/10.5169/seals-902500

CIP ist eine formale Entwicklungsmethode, mit der eingebettete Systeme durch grafische Modelle ablauffähig spezifiziert werden können. Der Entwicklungsansatz der Methode geht vom Verhalten der realen Objekte der Umgebung aus und führt in konstruktiven Schritten zum kompositionell definierten Gesamtsystem. Das auf die Methode zugeschnittene Spezifikationswerkzeug *CIP Tool* mit grafischen Editoren und C-Code-Generatoren erlaubt eine effiziente Anwendung der Methode in der Praxis.

Spezifizieren und Entwickeln mit grafischen Modellen

CIP – eine konstruktive Entwicklungsmethode für eingebettete Systeme

■ Hugo Fierz

In diesem Beitrag wird CIP (Communicating Interacting Processes) als Entwicklungsmethode für klassische eingebettete Systeme beschrieben. Der allgemeine Anwendungsbereich der Methode erstreckt sich sinngemäss auch auf Systeme, die Teil eines umfassenderen Softwaresystems sind, wie zum Beispiel die Leitprozesse eines Reglersystems oder die Protokolle einer bestimmten Schicht eines Kommunikationssystems.

Eingebettete Systeme, englisch Embedded Systems, sind Rechnersysteme, die als integrierender Bestandteil von Geräten und Anlagen für die Steuerung und Regelung der umgebenden Prozesse verantwortlich sind. In der Regel haben derartige Systeme speziellen Ansprüchen hinsichtlich Zuverlässigkeit und Einhalten von Echtzeitbedingungen zu genügen. Das notwendige ingenieurmässige Arbeiten wird durch die in der Praxis verbreiteten Case-Werkzeuge jedoch zuwenig unterstützt, da die verwendeten Methoden kaum durch formale Modelle abgestützt sind. Das Verhalten eines Systems kann mit solchen Methoden erst in der Implementationsphase durch die kompilierbaren Programme vollständig definiert werden. Die Folge sind oft Produkte, welche die hohen Qualitätsanforderungen an derartige Systeme nicht zufriedenstellend erfüllen, was sich unmittelbar in unnötig hohen Unterhaltskosten niederschlägt. Eine entscheidende Verbesserung der Softwarequalität ist möglich, wenn bereits in den problemorientierten Entwicklungsphasen mit formalen Prozessmodellen gearbeitet wird. Weil die Problemlösungen im Kontext der Anforderungen gefunden werden, entstehen naturgemäss weniger logische Fehler. Bekanntlich kommen solche Fehler besonders teuer zu stehen, wenn sie erst am implementierten System entdeckt werden. Zudem erlaubt die formalere Arbeitsweise den Entwicklungsprozess konsequenter durch Softwarewerkzeuge zu unterstützen, was eine weitere Senkung des Entwicklungsaufwandes mit sich bringt.

Eingebettete Systeme

Die Umgebung eines eingebetteten Systems (ES) besteht aus verschiedenen Anlageteilen mit eigenem aktivem Verhalten. Bei einfacheren Objekten ist dieses Verhalten in erster Linie durch deren physikalische Eigenschaften bestimmt. Komplexere Teile sind oft bereits durch

Dieser in Heft 17/95 publizierte Artikel wird nochmals abgedruckt, da bei der elektronischen Datenübergabe an die Druckerei eine grössere Anzahl von Figuren-Beschriftungen verlorengegangen sind.

Adresse des Autors:
Hugo Fierz, Dr. phys., Institut für Technische Informatik und Kommunikationsnetze, ETH-Zentrum, 8092 Zürich.

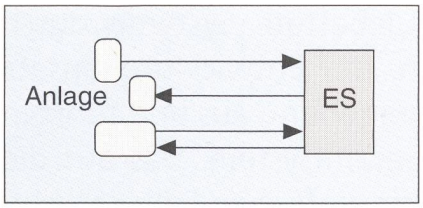


Bild 1 Eingebettetes System mit Umgebung

lokale Mikroprozessoren gesteuert (SPS, eingekaufte Produkte) und haben ein Verhalten, das wesentlich durch die integrierte Software bestimmt ist. Damit ein System durch den Menschen beeinflusst werden kann, muss die Anlage zusätzlich mit entsprechenden Bedienungsgeräten ausgestattet werden.

Die Koppelung zwischen Anlage und ES erfolgt über geeignete Wandler, die als Sensoren und Aktoren bezeichnet werden. Über Sensorsignale wird das ES über Zustandsänderungen der Anlage informiert, mittels Aktorsignalen kann auf die Anlage eingewirkt werden. Die allgemeine Aufgabe eines ES besteht darin, die einzelnen Anlagekomponenten so zu beeinflussen, dass ein bestimmtes Gesamtverhalten der Anlage entsteht.

Die CIP-Methode

Dieser Beitrag erläutert den Spezifikationsformalismus und die Entwicklungskonzepte der CIP-Methode [1]:

- *Operationeller Spezifikationsformalismus:* Das reaktive Verhalten eines eingebetteten Systems wird durch eine ausführbare Spezifikation formal vollständig beschrieben.
- *Umgebungsorientierter Entwicklungsansatz:* Die operationelle Spezifikation wird ausgehend vom Verhalten der realen Objekte der Umgebung in nachvollziehbaren Entwicklungsschritten gefunden.

An einem einfachen Fallbeispiel wird zum Schluss gezeigt, wie die vorgestellte Methode angewendet wird.

Operationelle Spezifikation eingebetteter Systeme

Der Entwickler eines eingebetteten Systems (ES) hat grundsätzlich zwei Aufgaben zu lösen:

- logisch richtige Reaktion des Systems auf Ereignisse der Umgebung
- Informationsübertragung zwischen ES und Umgebung

Diese beiden Aufgaben sollten möglichst unabhängig voneinander gelöst werden. Die logische Problemlösung kann als invarianter Teil verschiedener Implementationen eines eingebetteten Systems betrachtet werden und wird dementsprechend in einer höheren Abstraktionsebene beschrieben. Die logische Beschreibung, welche zum Beispiel das Verhalten der Steuerung eines Liftsystems festhält, wird keine Begriffe aus der Computertechnik verwenden. Von einer Lösung des logischen Problems kann jedoch nur gesprochen werden, wenn das Verhalten des eingebetteten Systems formal beschrieben wird. Das Problem der Informationsübertragung hingegen ist stark durch die verwendete Hardware geprägt und muss mit entsprechend speziellen Techniken gelöst werden.

Die vollständige Separierung der logischen Problemlösung von der Implementation der Informationsübertragung bringt grosse Vorteile im Entwicklungsprozess, da die beiden Aufgaben unabhängig voneinander gelöst werden können. Es handelt sich hier nicht einfach um die Zerlegung eines grossen Problems in zwei kleinere, sondern es wird die Entflechtung zweier Problemkreise erreicht, welche verschiedenen Abstraktionsebenen angehören.

Eine *operationelle Spezifikation* [2] ist die konstruktive Beschreibung des Verhaltens eines Systems durch eine abstrakte Maschine (Petrinetze [4], Zustandsmaschinen [3], sequentielle Prozesse [5]). Dabei bedeutet operationell, dass das zugrunde liegende Modell logisch ablauffähig ist. Geeignete Softwarewerkzeuge ermöglichen, aus so spezifizierten Modellen ablauffähigen Code zu erzeugen, der direkt für Simulationen oder für die Implementation auf dem Zielsystem verwendet werden kann. Aus denselben Modellen lässt sich auch die Dokumentation erzeugen, die den gleichen formalen Gehalt hat wie der erzeugte Code.

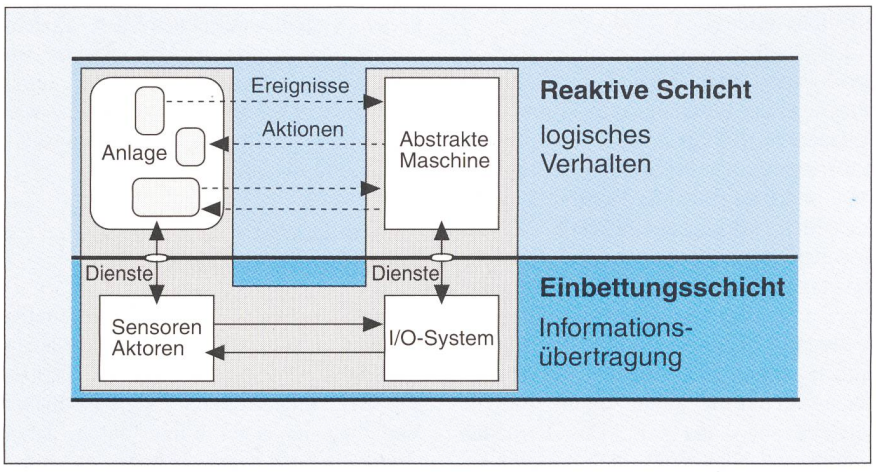


Bild 2 Abstraktionsebenen eingebetteter Systeme

Der operationelle Spezifikationsformalismus von CIP

Bei der CIP-Methode wird das Systemverhalten durch kooperierende Zustandsmaschinen spezifiziert. Ein CIP-System besteht aus asynchron kooperierenden Clustern (true concurrency). Jeder Cluster setzt sich aus synchron kooperierenden Prozessen zusammen, die als erweiterte Zustandsmaschinen (Automaten) beschrieben werden. Das mathematische Modell eines Clusters ist als Produkt dieser Zustandsmaschinen definiert und stellt eine Zustandsmaschine mit mehrdimensionalem Zustandsraum dar; das heisst, der Zustand eines Clusters ist durch die Zustände seiner Prozesse gegeben.

Durch ein Ereignis aus der Umgebung wird ein Zustandsübergang des empfangenden Prozesses erzeugt. Der so aktivierte Prozess kann einen Puls erzeugen und damit weitere Prozesse des Clusters anstossen, die wiederum mittels Pulsen weitere Prozesse aktivieren können. Die durch Pulsübertragung entstehende Kettenreaktion ist ununterbrechbar und definiert einen einzigen Zustandsübergang des ganzen Clusters. Mittels Aktionen wirken die aktivierten Prozesse auf die Umgebung zurück.

Die Prozesse der verschiedenen Cluster können über Datenströme asynchron kommunizieren. Asynchrone Kommunikation bedeutet im CIP-Formalismus, dass die Schreib- und die Leseaktion einer Übertragung zeitlich getrennt in zwei verschiedenen Clustertransitionen ausgeführt werden.

Prozesse:
Erweiterte Zustandsmaschinen

Prozesse werden als erweiterte Zustandsmaschinen spezifiziert: Mit Transitionsstrukturen und durch die in den Transitionen ausgeführten Operationen kann in

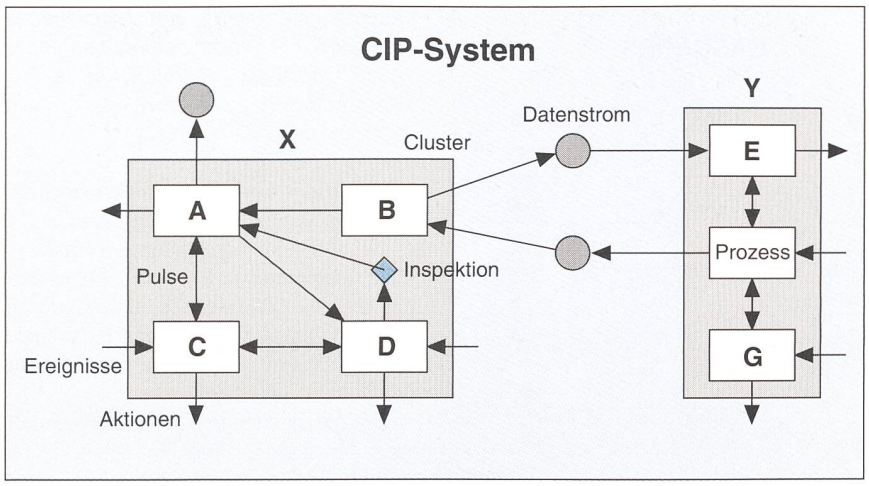


Bild 3 CIP-System mit zwei Clustern

zwei verschiedenen Abstraktionsebenen Funktionalität spezifiziert werden. Die Transitionsstruktur beschreibt die Kooperation mit anderen Prozessen und mit den Objekten der Umgebung; mit den Operationen wird die lokale algorithmische Funktionalität definiert.

Die in Bild 4 abgebildete Transitionsstruktur definiert das Verhalten des Prozesses *Einnehmer* eines Verkaufsautomaten. Der Prozess befindet sich immer in einem der durch Kreise dargestellten Zustände. Das externe Ereignis *Muenze* oder die Pulse *abbruch* und *verkauft* können einen Zustandsübergang auslösen. In jeder Transition ist es möglich, die Aktion *Auswurf* und einen neuen Puls zu erzeugen.

Für algorithmische Belange können in jedem Prozess statische Variablen und Datentypen für Ereignisse, Pulse, Aktionen und Meldungen deklariert werden. Die Algorithmen werden mittels Operationen

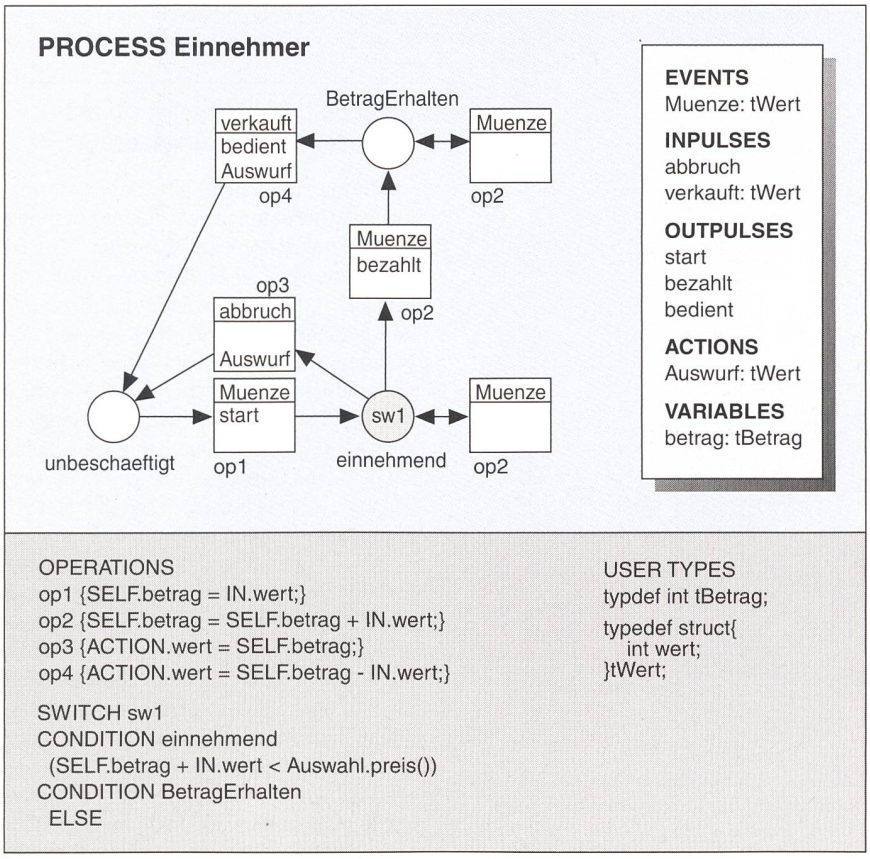


Bild 4 Prozess Einnehmer eines Verkaufsautomaten

spezifiziert, die in den Zustandsübergängen ausgeführt werden.

Im grau markierten Zustand *einnehmend* des Prozesses Einnehmer in Bild 4 sind für das Ereignis *Muenze* zwei Transitionen möglich (Nondeterminismus). Durch entsprechende Bedingungen des zugeordneten Switch *sw1* wird das Prozessverhalten eindeutig. Die Bedingungen eines solchen Switch können von den Daten des Ereignisses und den Werten der eigenen Variablen sowie von den Zuständen und Variablen anderer Prozesse desselben Clusters abhängen (siehe Zustandsinspektion).

Variablen, Datentypen, Operationen und Bedingungen werden in CIP in der Programmiersprache C oder C++ formuliert. Die Code-Fragmente werden im generierten C-Code «inline» eingebunden. Vom theoretischen Standpunkt aus wäre es schöner, eine eigene funktionale Sprache zu verwenden. In der Praxis hat sich jedoch die Nützlichkeit des pragmatischen Ansatzes mit ANSI-C bzw. C++ eindeutig bestätigt, erlaubt er doch problemlos Funktionen, Datentypen und Objektklassen bestehender Bibliotheken zu verwenden.

Eingebettete Systeme haben im allgemeinen einen hybriden Charakter, indem die Erfassung des Anlageverhaltens teils über diskrete Sensoren, teils durch quasikontinuierliche Abtastung erfolgt. Für die Regelung der abgetasteten Vorgänge werden in CIP datentragende Abtastereignisse und Stellaktionen verwendet; die regeltechnischen Algorithmen werden in Operationen der aktivierten Transitionen ausgeführt.

Interaktion: Synchrone Pulsübertragung, Zustandsinspektion

Die synchrone Übertragung von Pulsen wird als Interaktion bezeichnet. Durch ein *Interaktionsnetz* (Bild 5) wird spezifiziert, zwischen welchen Prozessen Pulse übertragen werden können.

Die Struktur des Interaktionsnetzes schränkt die möglichen Pulsübertragungsketten jedoch zuwenig ein, da im allgemeinen zyklische Übertragungswege möglich sind. Damit diese Kettenreaktionen eindeutig und beschränkt definiert sind, muss für jeden Prozess, der Ereignisse oder Meldungen erwartet, die durch ihn aktivierbare Prozesskaskade definiert werden. Eine *Kaskade* ist ein baumartiges Subnetz des Interaktionsnetzes, in welchem der oberste Prozess durch einen externen Input aktiviert wird. Erzeugt ein in einer Kaskade aktivierter Prozess keinen Puls oder erwartet ein Empfänger einen erzeugten Puls nicht, so bricht die Ausführung der Kaskade in diesen Knoten ab.

Wie bereits erwähnt, können die Bedingungen einer Transitionsstruktur direkt von

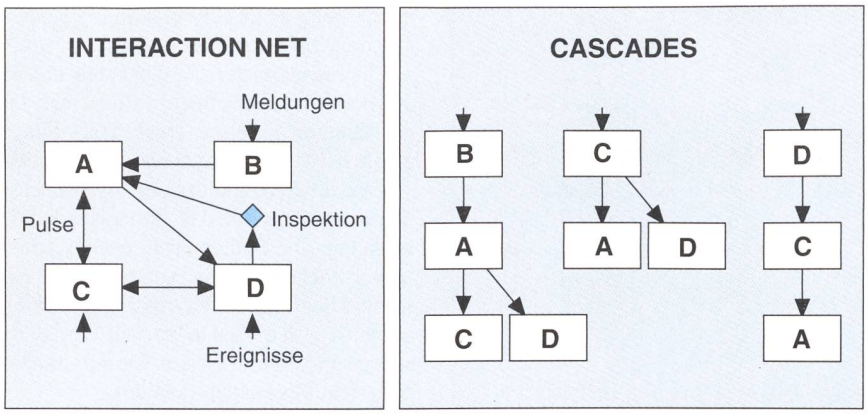


Bild 5 Interaktionsnetz und Kaskaden eines Clusters

den Zuständen und Variablen anderer Prozesse desselben Clusters abhängen. Der Lesezugriff auf die Daten eines anderen Prozesses wird als Zustandsinspektion bezeichnet und erfolgt wie in objektorientierten Modellen über Zugriffsfunktionen (read-only methods) des inspizierten Prozesses. Die gelieferte Information bezieht sich immer auf die Zustände und Variablenwerte unmittelbar vor der aktuellen Clusteraktivierung. Im Gegensatz zur Pulsübertragung, bei welcher sowohl Sender wie Empfänger aktiv an der Übertragung teilnehmen, wird bei einer Inspektion der inspizierte Prozess nicht aktiviert.

Durch Zustandsinspektion entsteht eine weitere Abhängigkeit zwischen Prozessen, die im Interaktionsnetz durch Verbindungen mit Diamanten grafisch deklariert werden (Bild 5). Die Pfeile geben die Richtung des Datenflusses an.

Kommunikation: Asynchrone Übertragung von Meldungen

Die Informationsübertragung zwischen Prozessen verschiedener Cluster wird durch asynchrone Kommunikation mittels Datenströmen spezifiziert, wobei auch Prozesse desselben Clusters mit Datenströmen verknüpft werden können. Datenströme sind sequentielle Meldungspuffer und modellieren das Kommunikationsmedium des verteilten Systems. Wie bei der Interaktion wird die Datenflussstruktur durch ein grafisches Netzmodell spezifiziert. Die kommunizierenden Prozesse werden dabei über Outports und Inports mit Datenströmen verknüpft. Für jeden Outport und Inport werden Meldungen definiert, die in Prozesstransitionen geschrieben und gelesen werden können.

Moderation: Verhaltensstrukturierung

Durch verschiedene Modi können für einen Prozess verschiedene Verhaltensmuster definiert werden. Jeder Modus wird durch eine Transitionsstruktur wie in Bild 4 beschrieben, welche die Zustände, Ereignisse, Pulse, Aktionen und Meldungen des Prozesses verwendet.

Der aktive Modus eines Prozesses definiert das aktuelle Prozessverhalten. Durch eine weitere Transitionsstruktur (Moderator) auf den Modi des Prozesses wird spezifiziert, wie das Verhalten des Prozesses gewechselt werden kann. Modustransitionen werden durch spezielle Pulse erzeugt. Bei einer Modustransition bleibt der aktuelle Prozesszustand erhalten; es ist jedoch möglich, in einer Modustransition einen Trigger abzusetzen, der im neuen Modus sofort einen ersten Zustandsübergang erzeugt.

Die Strukturierung des Prozessverhaltens durch verschiedene Modi hat sich bewährt; es lassen sich damit die direkte Beeinflussung eines Prozesses von der Beeinflussung des Verhaltens separieren. Das Strukturierungsmittel erlaubt, auf einfache Weise die Komplexität reaktiver Systeme durch Verhaltenshierarchien in den Griff zu bekommen.

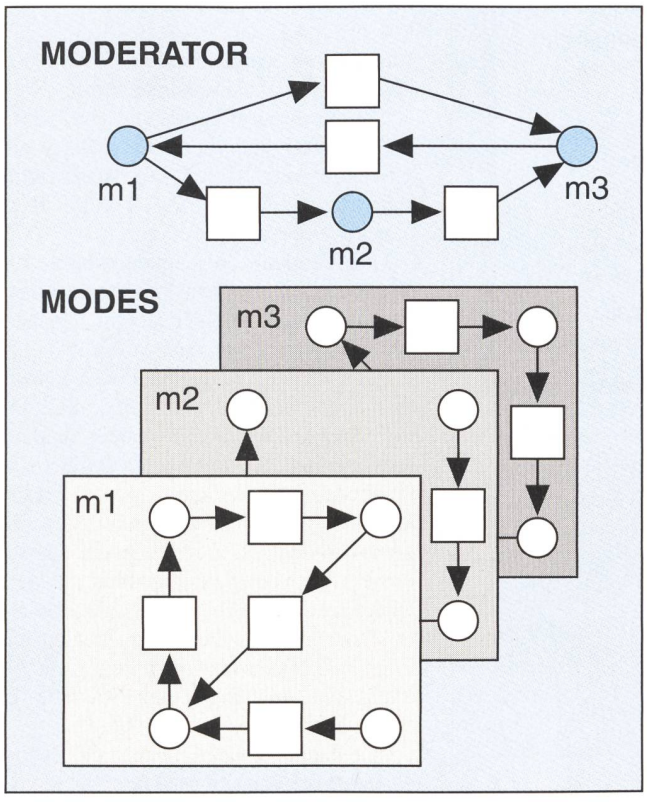


Bild 6 Drei Modi mit Moderator

Umgebungsorientierter objektbasierter Entwicklungsansatz

Mit operationellen Spezifikationsformalismen wird automatisch in einer definierten Abstraktionsebene gearbeitet. Zusätzlich zu solchen «Leitplanken» sollte eine Methode für den Entwicklungsprozess ein konzeptuelles Modell vorschlagen, mit welchem in nachvollziehbaren Schritten verständliche Lösungen gefunden werden können. Ein «richtig laufendes System» ist noch kein Garant für Qualität. Entscheidend ist, dass eine Strukturierung gefunden werden kann, die auch einen sicheren Unterhalt des Systems ermöglicht.

Bei der operationellen Spezifikation mit der CIP-Methode wird vom Verhalten der realen Objekte der Systemumgebung ausgegangen. Der Ansatz stammt aus der JSD-Methode (Jackson System Development) [5] und wird auch in den neueren objektorientierten Analysemethoden angewendet. Das realitätsorientierte Vorgehen führt zu verständlichen Systemen, was sich unmittelbar auf Qualitätsmerkmale wie Zuverlässigkeit, Wartbarkeit und Änderungsfreundlichkeit auswirkt.

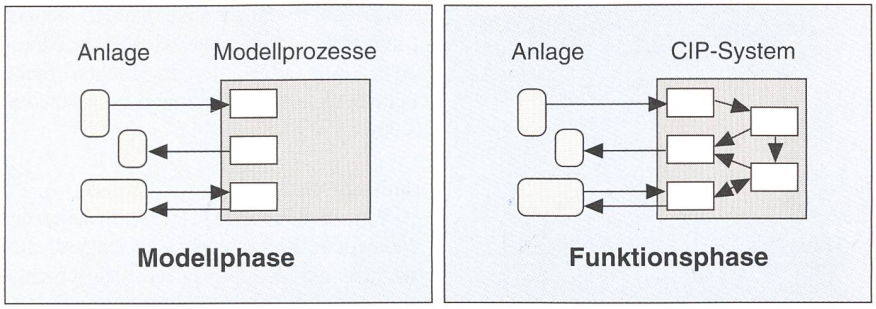


Bild 7 Entwicklungsphasen der CIP-Methode

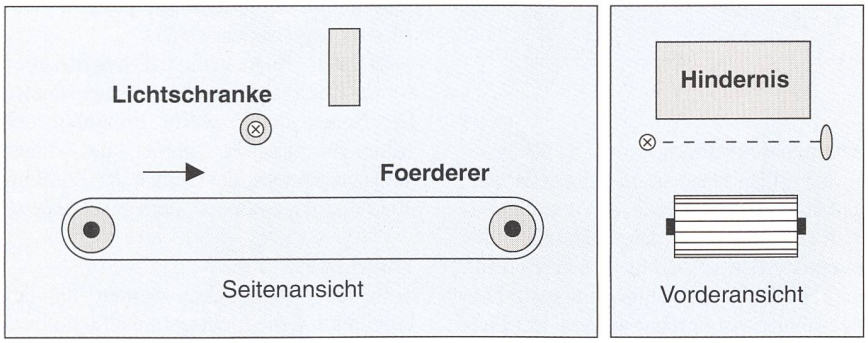


Bild 8 Anlage des Fallbeispiels

Wir beschreiben in diesem Beitrag die Modell- und die Funktionsphase für die Entwicklung eines einzigen Clusters (Bild 7). Bei Systemen mit mehreren Clustern käme noch eine Entwicklungsphase für die globale Kommunikation dazu.

Modellphase

Als erste Komponenten eines CIP-Systems spezifiziert man pro Objekt der Umgebung einen sogenannten Modellprozess, welcher die Sensorsignale des Objektes als Ereignisse empfängt und durch Aktionen entsprechende Aktorsignale erzeugen kann. Die Transitionsstrukturen dieser Prozesse beschreiben, in welchen Folgen Ereignisse auftreten und Aktionen erzeugt werden können (sequentielle Protokolle). Das spezifizierte System enthält so explizite Modelle, die das gültige Verhalten der Anlageteile aus der Sicht des eingebetteten Systems definieren. Die Modellprozesse bilden damit die Grundlage für die Entwicklung der reaktiven Funktionalität des Systems.

Funktionsphase

In der zweiten Entwicklungsphase wird das Gesamtverhalten des eingebetteten Systems spezifiziert. Dazu werden Funktionsprozesse eingeführt und mittels Pulsübertragung und Statusinspektion die notwendigen Abhängigkeiten zwischen den Modellprozessen definiert. In einem ersten Schritt entwickelt man die primäre Funktionalität des Systems, die sich auf das durch die Modellprozesse definierte Normalverhalten abstützt. Damit auch auf ungültiges Verhalten der Umgebung und auf Übertragungsfehler reagiert werden kann, müssen meistens in weiteren Schritten die Strukturen der Modellprozesse erweitert und Überwachungsprozesse eingeführt werden. Wie die Erfahrung zeigt, gehört die Spezifikation des Systemverhaltens für den Fehlerfall zu den schwierigsten Problemen bei der Entwicklung robuster und sicherer Systeme. Ohne explizites Modell des Normalverhaltens kann diese Aufgabe meist gar nicht oder zumindest nicht befriedigend gelöst werden.

Ein weiterer Vorteil des kompositionellen Ansatzes zeigt sich, wenn Änderungswünsche realisiert werden müssen. Modelle der realen Umgebung werden sich in den meisten Fällen als stabiler erweisen, als eine durch funktionale Dekomposition gefundene Systemstruktur, die auf einem speziellen Katalog funktionaler Anforderungen basiert.

Kleine Anwendung

An einem einfachen Beispiel werden die Entwicklungsschritte der CIP-Methode erläutert. Die resultierende Spezifikation be-

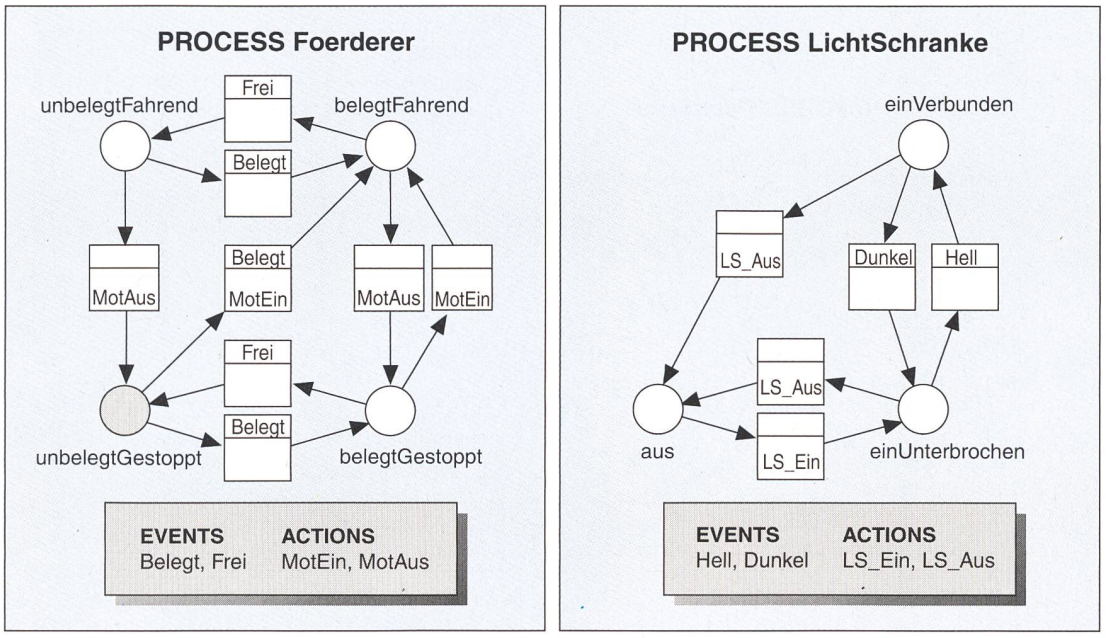


Bild 9 Modellprozesse mit Ereignissen und Aktionen

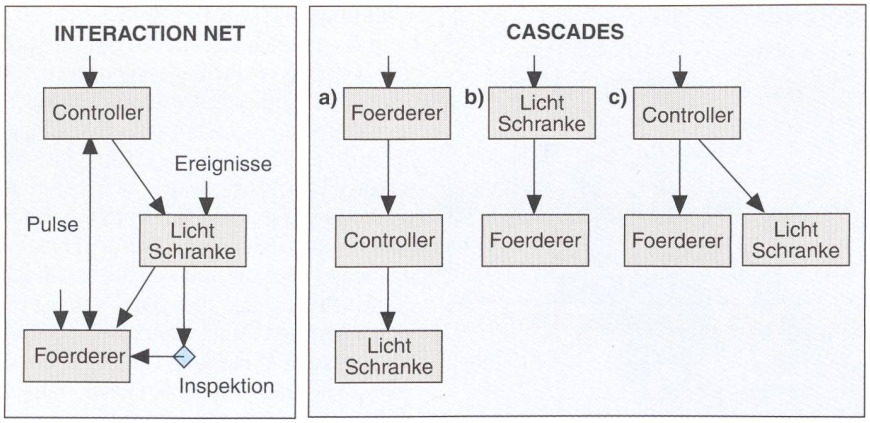


Bild 10 Interaktionsnetz und Kaskaden

schreibt formal vollständig die synchrone Kooperation von drei CIP-Prozessen. Die algorithmischen Anforderungen in diesem Beispiel sind trivial, das heisst, die Prozesse haben keine Variablen, führen keine Operationen aus und es werden keine Daten übertragen.

Anlagebeschreibung

Die Anlage besteht aus einem Förderer und einer Lichtschranke, mit der die maximal erlaubte Höhe des Förderguts überwacht werden soll (Bild 8). Die Belegung des Förderers kann über einen digitalen Belastungssensor erfasst werden. Der Lichtsensor der Lichtschranke meldet, ob der Lichtstrahl empfangen wird. Der Motor des Förderers und die Lichtschranke können elektronisch ein- und ausgeschaltet werden.

Anforderungen

Wenn der Förderer belegt wird, sollen der Motor und die Lichtschranke eingeschaltet werden. Bei unterbrochener Lichtschranke darf der Motor des Förderers nie laufen. Ist der Förderer während 30 Sekunden unbelegt, so sollen Förderer und Lichtschranke wieder ausgeschaltet werden.

In der folgenden CIP-Spezifikation wurde folgende Schreibregel angewendet: Die Namen der Prozesse, Ereignisse und Aktionen beginnen immer mit einem Grossbuchstaben, die Namen der Zustände und Pulse dagegen sind klein geschrieben.

Modellphase

In der Modellphase werden für den Förderer und die Lichtschranke Modellpro-

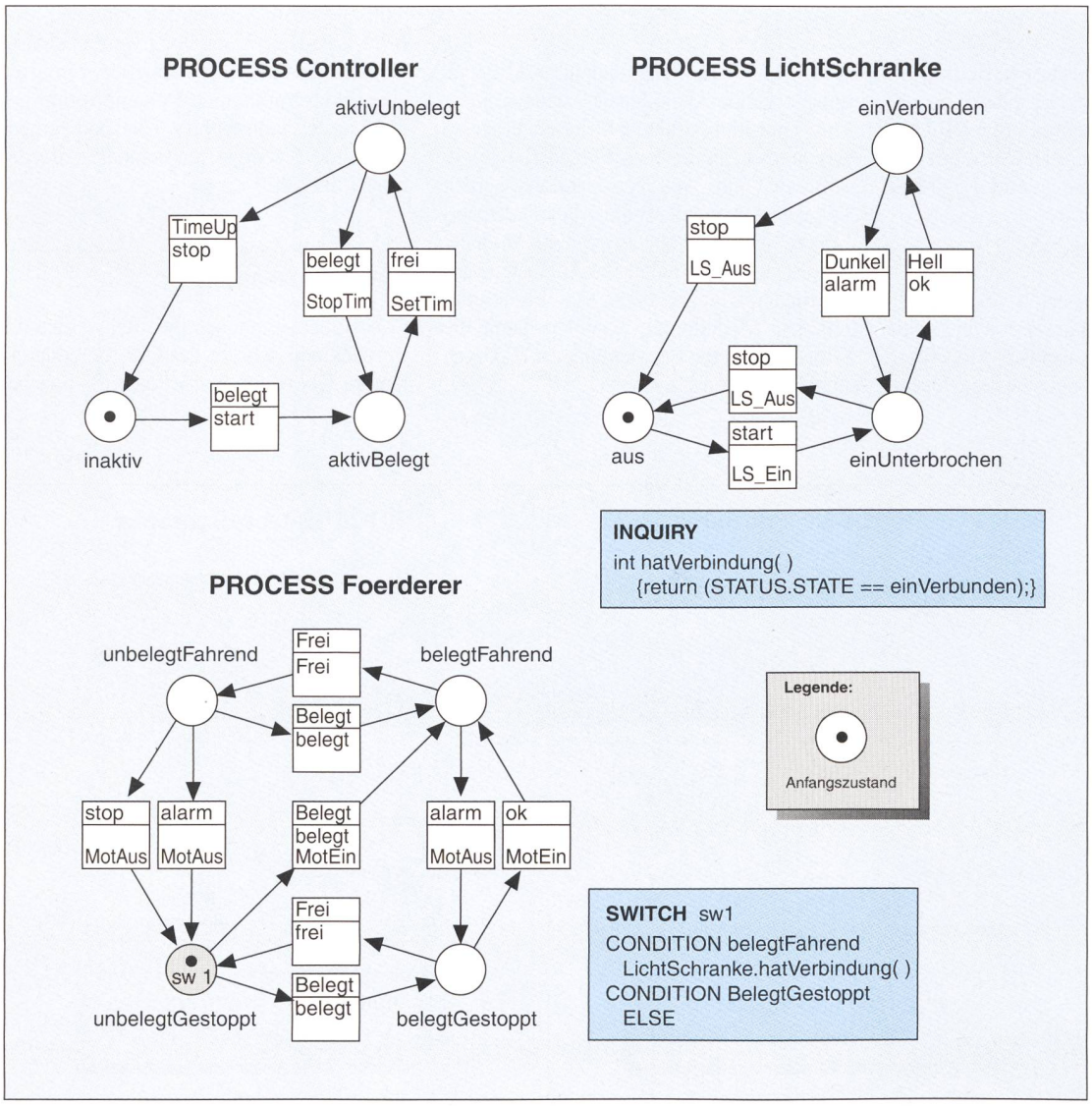


Bild 11 Vollständige Transitionsstrukturen

Zur Projektgeschichte von CIP

Die ersten Ansätze zur CIP-Methode sind 1989 an der Software-Schule Schweiz in Bern entstanden. Anschliessend sind in einem dreijährigen KWF-Projekt (Kommission zur Förderung der wissenschaftlichen Forschung, Projekt-Nrn. 2112, 2268) am TIK der ETH Zürich die formalen Grundlagen der Methode ausgearbeitet und ein erstes Spezifikationswerkzeug entwickelt worden. Das KWF-Projekt wurde vom Bund, von der Industrie (Ascom Autelca, EPS, Hasler-Stiftung, Landis & Gyr, Securiton) und von der durch die Projektgruppe gegründeten Firma CIP System AG getragen. Die Tauglichkeit der Methode und des Werkzeuges ist in Pilotprojekten der industriellen Projektpartner eindrücklich bestätigt worden. In der Folge hat die CIP System AG das Resultat des KWF-Projektes zum kommerziellen Produkt CIP Tool weiterentwickelt. Das Werkzeug wird bereits im Unterricht (ETH Zürich, NDIT/FPIT, HTL Bern, Biel und Rapperswil) sowie in verschiedenen industriellen Projekten eingesetzt. Seit Herbst 1994 besteht eine Zusammenarbeit mit der Firma Zühlke Engineering AG, die als Kompetenzzentrum für die CIP-Methode Grundkurse und projektbegleitende Unterstützung anbietet.

Die CIP-Methode wird in verschiedenen Richtungen weiterentwickelt. Die geplanten Forschungsarbeiten an der ETH Zürich betreffen die formale Modellierung von Verhaltenshierarchien, die Codegenerierung für Multiprozessorsysteme und die formale Verifikation von Systemeigenschaften.

zesse (Bild 9) mit Ereignissen und Aktionen spezifiziert.

Der Belastungssensor erzeugt die Ereignisse *Belegt* und *Frei* für den Prozess *Foerderer*, der mit den Aktionen *MotEin* und *MotAus* den Motor steuern kann. Der Lichtsensor erzeugt die Ereignisse *Hell* und *Dunkel* für den Prozess *LichtSchranke*. Mit den Aktionen *LS_Ein* und *LS_Aus* kann dieser die Lichtschranke ein- und ausschalten.

Die Struktur des Prozesses *Foerderer* enthält eine nondeterministische Verzweigung: Beim Belegen des gestoppten Förderers hängt die Reaktion des Modellprozesses vom Zustand der Lichtschranke ab. Dies wird durch zwei verschiedene Transitionen für das Ereignis *Belegt* ausgedrückt, beide ausgehend vom grau markierten Zustand *unbelegtGestoppt*.

Die Struktur des Prozesses *LichtSchranke* berücksichtigt, dass nach dem Einschalten als erstes das Ereignis *Hell* auftritt.

Funktionsphase

In der Funktionsphase wird zusätzlich der Steuerprozess *Controller* eingeführt, der für die Aktivierung und Deaktivierung des Fördersystems verantwortlich ist (Bild 11). Der Prozess benützt einen externen Timer, der mit den Aktionen *SetTim* und *StopTim* gesetzt und gestoppt werden kann. Mit dem Ereignis *TimeUp* meldet sich der abgelaufene Timer zurück. Interne Timer stehen in CIP ebenfalls zur Verfügung.

Alle drei Prozesse bilden einen einzigen Cluster. Das geforderte Verhalten des Systems entsteht durch gezielte Pulsübertragung. Durch das *Interaktionsnetz* (Bild 10) wird festgelegt, zwischen welchen Prozessen Pulse übertragen werden dürfen. Die Verbindung mit dem Diamanten definiert zudem, dass der Prozess *Foerderer* den Prozess *LichtSchranke* inspizieren kann.

Beim Entwickeln der Kaskaden haben folgende Szenarien eine Rolle gespielt:

a. Die Ereignisse *Belegt* und *Frei* des Förderers werden dem Controller gemeldet, der dafür sorgt, dass die Lichtschranke eingeschaltet wird, und der den Timer für das Ausschalten setzt.
b. Die Ereignisse der Lichtschranke müssen eine Wirkung auf den Förderer haben können.
c. Wenn der Timer des Controllers abläuft, müssen Förderer und Lichtschranke ausgeschaltet werden.

Welche Pulse wann übertragen werden, ist in den vervollständigten Transitionsstrukturen der Prozesse definiert (Bild 11).

Die nondeterministische Verzweigung für das Ereignis *Belegt* im Zustand *unbelegtGestoppt* des Prozesses *Foerderer* wird durch den zugeordneten Switch *sw1* eindeutig ausführbar. Die Bedingung für den Folgezustand *BelegtFahrend* enthält den Aufruf der Funktion *hatVerbindung()*, welche Information aus dem aktuellen Kontext des Prozesses *LichtSchranke* liefert (Zustandsinspektion).

Das Verhalten des Systems ist durch die zeitliche Folge der auftretenden äusseren Ereignisse bestimmt. Durch ein Ereignis eines Prozesses wird die entsprechende Prozesskaskade ausgeführt. Ein möglicher Ablauf des Systems (Trace) könnte mit folgenden drei Clustertransitionen beginnen:

Ereignis *Belegt* für *Foerderer*	-> Zustand *belegtGestoppt*
– Puls *belegt* für *Controller*	-> Zustand *aktivBelegt*
– Puls *start* für *LichtSchranke*	-> Zustand *einUnterbrochen*, Aktion *LS_Ein*
Ereignis *Hell* für *LichtSchranke*	-> Zustand *einVerbunden*
– Puls *ok* für *Foerderer*	-> Zustand *belegtFahrend*, Aktion *MotEin*
Ereignis *Frei* für *Foerderer*	-> Zustand *unbelegtFahrend*
– Puls *frei* für *Controller*	-> Zustand *aktivBelegt*, Aktion *SetTim*

Literatur

[1] *H. Fierz, H. Müller, S. Netos:* CIP-Communicating Interacting Processes. In *J. Gorski* (ed.): Proceedings Safecomp'93, Poznan-Kiekrz, Poland, 1993. Springer Verlag 1993.

[2] *P. Zave:* The Operational Approach versus the Conventional Approach to Software Development. Comm. ACM 27 (1984) 2, pp. 104–118.

[3] *D. Harel:* Statecharts: A Visual Formalism for Complex Systems. Science of Computer Programming (1987) 8, pp. 231–274.

[4] *R. Mattman:* Modellieren mit Specs-Netzen. Bulletin SEV/VSE 86(1995)9, pp. 11–16.

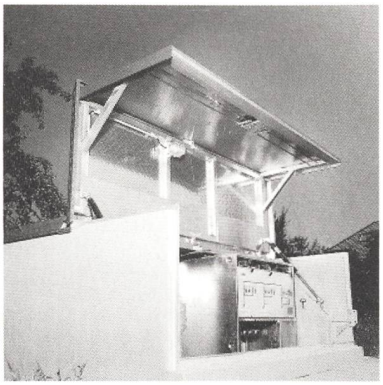
[5] *J.R. Cameron (ed.):* JSP and JSD: The Jackson Approach to Software Development. IEEE Computer Society Press, 1989.

CIP – une méthode formelle de développement pour des systèmes de commande de processus

CIP est une méthode formelle de développement qui permet de spécifier des systèmes de commande de processus exécutables à l'aide de modèles graphiques. L'approche méthodologique part du comportement des objets réels de l'environnement et mène pas à pas de façon constructive à une spécification exécutable du système. Avec l'outil de spécification *CIP Tool*, ses éditeurs graphiques et générateurs de code C, la méthode s'applique en pratique de manière efficace.

Après une introduction à l'approche opérationnelle appliquée aux systèmes de commande de processus, l'article présente le formalisme graphique de CIP. Puis, on explique le processus de développement proposé par CIP. Un exemple complet illustre la théorie.

TRETEN SIE MAL IN IHRE STROMVERSORGUNG



Bitte einzutreten: Mit unserer begehbaren Kompaktstation T 85 wird bei der Wartung die Tür zum Dach. So sparen Sie teure Aufstellfläche, die Sie bisher zum Öffnen der Türen einplanen mussten. Und mit einer Bauhöhe über Terrain von nur 1m sorgt diese unauffällige Station für freies Blickfeld an Kreuzungen und vor Wohnfenstern. Die T 85 ist eine unserer vier Kompaktstationstypen, mit denen Sie viele Ihrer Planungs- und Versorgungsaufgaben lösen. Alle Stationsgeometrien sind fugenlos aus einem Guss fabrikgefertigt nach System Betonbau, bestehen aus 100 mm Stahlbeton B 35, bieten vorbereitete BBK-Kabeldurchführungen und sind optisch perfekt anpassbar an Ihre jeweilige Bauumgebung. Eines unserer vier Werke ist in Ihrer Region. Nähere Informationen bei Betonbau GmbH, Postfach 1161, 68743 Waghäusel, Tel. (0 72 54) 9 80 - 4 01, Fax (0 72 54) 9 80 - 4 19.

BETONBAU GMBH

Qualität aus einem Guss.

PKG

PKG-versicherte Firmen haben gut lachen

PKG Der GAV-konforme Branchenkenner

PKG Die Kranken-versicherung mit den stabilen Prämien

PKG Auch für Ihre Firma! **Vergleichen kostet nichts...**

...kann Ihrer Firma aber sehr viel bringen!
Wir beraten Sie gerne.

PKG

PKG
Paritätische Krankenversicherung
für Branchen der Gebäudetechnik
Postfach 272
3000 Bern 15

Telefon 031 / 350 24 24
Telefax 031 / 350 22 33

PS: PKG – die Krankenversicherung der Verbände SSIV, VSEI, VSHL und SMUV mit mehr als 900 angeschlossenen Firmen.