Netzbasierte Softwaretechnik in Verteilten Systemen

Prof. Dr. Heinrich P. Godbersen

Technische Fachhochschule Berlin, Fachbereich Informatik, Luxemburger Straße 10, D 13353 Berlin

---

Zusammenfassung

Dieser Beitrag beschreibt eine graphisch orientierte Umgebung zur Entwicklung Verteilter Systeme.

---

1 Einleitung

Mit zunehmender Verbreitung von Rechnernetzen wird die Entwicklung verteilter Systeme immer wichtiger. Einen Ansatz, die Komplexität dieser Aufgabenstellung zu reduzieren und dabei gleichzeitig den Erstellungsprozeß zu flexibilisieren und effizienter zu gestalten, bietet die Software-Produktionsumgebung VerSoS. Ziel von VerSoS ist es, eine Umgebung zu schaffen, in der verteilte Anwendungen auf einer graphischen Oberfläche spezifiziert, synthetisiert und unmittelbar ausgeführt werden können. Das homogen konzipierte System basiert auf der Methode der Funktionsnetze, einer Petrinetzklasse. Die Software-Erstellung auf der Grundlage des phasenübergreifenden Funktionsnetz-Ansatzes bietet eine Reihe von Vorteilen, beispielsweise die intuitiv verständliche und präzise graphische Darstellung von Systemen, eine integrierte Dokumentation, evolutionäres Prototyping durch den Gebrauch operationaler Modelle und die Wiederverwendung von Software-Bausteinen. Der

Notwendigkeit, "maßgeschneiderte Standardsoftware" zu produzieren, wird damit Rechnung getragen.

Nach einer einführenden Vorstellung der eingesetzten Methode werden die Eigenschaften des gewählten Ansatzes aus der Sicht des Software-Engineering aufgezeigt. Im nächsten Teil wird die Software-Produktionsumgebung in einer Systemübersicht vorgestellt. Es folgt eine kurze Beschreibung der Realisierungsgrundlage für die verteilte Ausführung und eine Erläuterung, wie Teile von VerSoS selbst umgesetzt wurden. Ein Beispiel zur Nutzung der vorgestellten Umgebung und Bemerkungen zu VerSoS schließen den Beitrag ab.

zurück

---

2 Vorstellung der methodischen Grundlage

Funktionsnetze [l] bauen auf der Kanal-/Instanz-Interpretation von Petrinetzen auf. Durch die Erweiterung der sprachlichen Ausdrucksmittel, z. B. durch Zusammenfassung von Teilnetzen zu eigenen Symbolen, erleichtern Funktionsnetze die Modellierung von Systemen. Abbildung 1 zeigt die Eigenschaften der für VerSoS erweiterten Funktionsnetze in einer Übersicht.

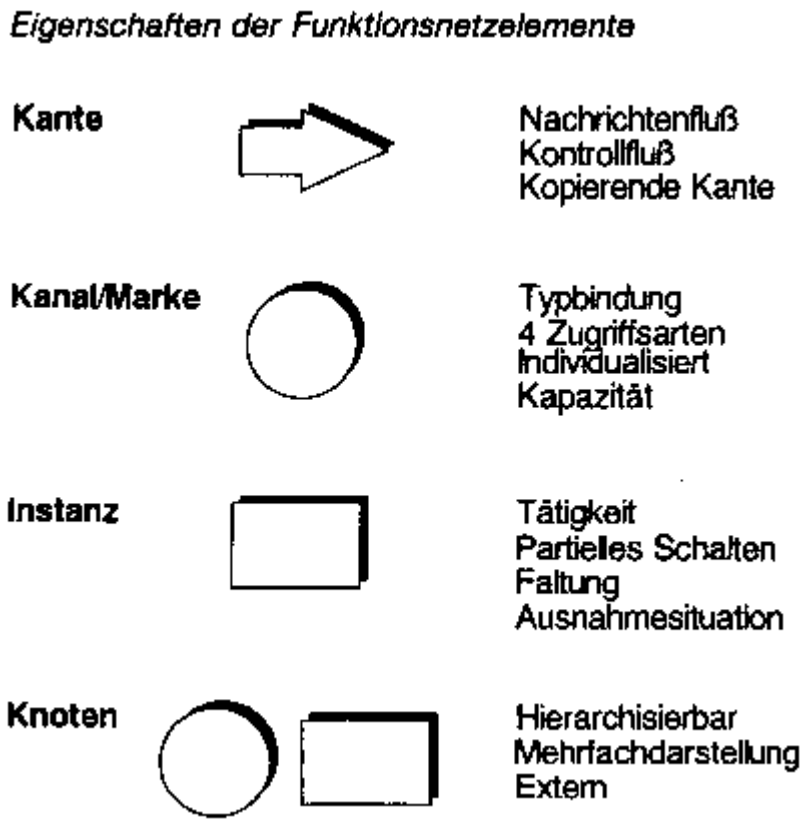

Abbildung 1: Eigenschaften der Netzelemente

Die Kennzeichnung *externer* Knotenelemente ist eingeführt worden, um - im Falle von Kanälen - die Einbindung externer Datenbestände (z.B. Datenbank) und - im Falle von Instanzen - fertiger Software zu beschreiben. Funktionsnetze bieten die Möglichkeit,

vorhandene (teure) Software, die u. U. schon erfolgreich im Einsatz ist, zu neuen verteilten Anwendungen zusammenzufassen. Auch Programme, deren Quellen nicht zur Verfügung stehen, können mit diesem Mechanismus eingebunden werden.

zurück

---

3 Unterstützung des Software-Engineering

Der Funktionsnetz-Ansatz unterstützt Aspekte des Software-Engineerings für die Synthese verteilter Software. Die wichtigsten Gesichtspunkte sind:

- Graphische Spezifikation,
- Evolutionäres Prototyping,
- Einheitliche Schnittstellen des Werkzeugverbundes,
- Effizienzsteigerung durch Wiederverwenden von Software-Bausteinen,
- Integrierte Dokumentation und
- Nebenläufigkeit im Entwicklungsprozeß.

zurück

---

4 Übersicht über das VerSoS-System

In VerSoS werden Werkzeuge und Datenbestände unterschieden. Werkzeuge repräsentieren Tätigkeiten und werden deshalb bei der Darstellung als Netz durch Instanzen (eckige Knotenelemente) symbolisiert. Daten werden als Kanäle (runde Knoten) abgebildet. Abbildung 2 zeigt das VerSoS-System im Überblick. Einzelne Knoten können verfeinert sein.

Verschiedene Werkzeuge, die den gesamten Lebenszyklus einer verteilten Anwendung instrumentell unterstützen, sind über die Darstellung als Funktionsnetz verbunden.

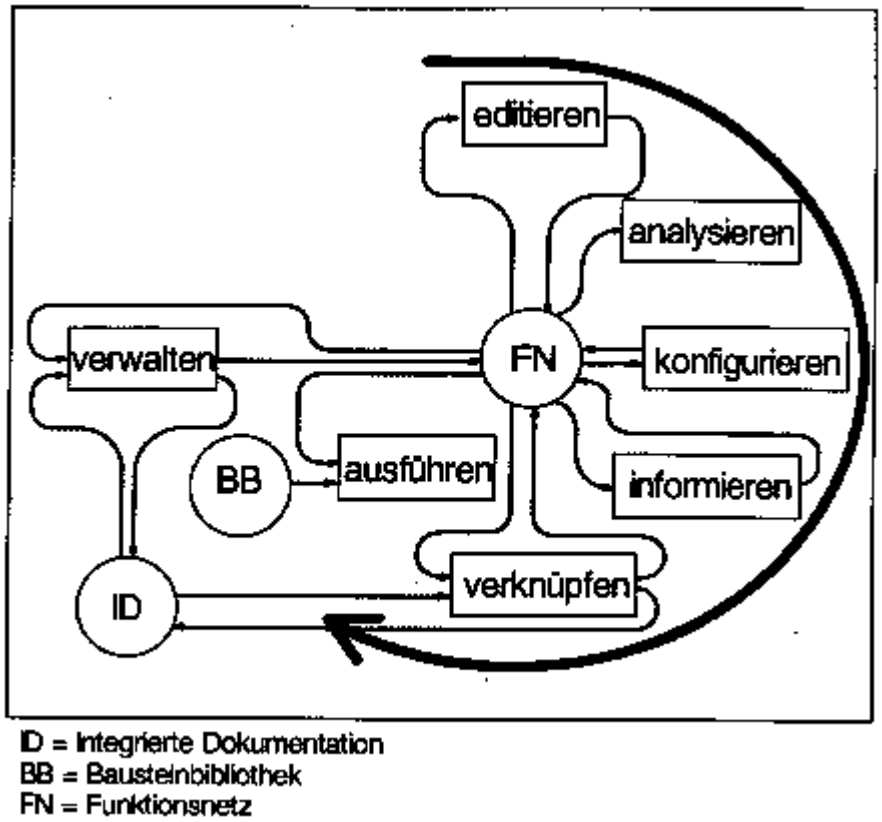

Abbildung 2*:* Systemmodell von VerSoS

zurück

---

5 Vorgehensweise bei der Entwicklung verteilter Anwendungen

Ein wesentliches Ziel bei der Gestaltung des VerSoS-Systems ist es, eine Möglichkeit zu schaffen, von den programmiertechnischen Details der Verteilung zu abstrahieren.

Die notwendigen Schritte bei der Erstellung einer verteilten Anwendung sind die Modularisierung und die Verteilung eines Gesamtsystems. In VerSoS wird dies so gelöst: Eine verteilte Anwendung wird entworfen (Funktionen "Editieren", "Konfigurieren", vgl. Abbildung 2). Die Knotenelemente werden hinreichend spezifiziert und mit Bausteinen verbunden (Funktion "Verknüpfen"). Für die Ausführung des Modells werden nun von einer Ausführungskomponente, die dem Anwender verborgen bleibt, die Bausteine aus der Bibliothek extrahiert und gemäß den Konfigurationsvorgaben auf die Rechner verteilt. Die Ausführungskomponente ist außerdem für den Aufbau von Kommunikationsbeziehungen (Datenflüsse zwischen Bausteinen) gemäß der Modellspezifikation verantwortlich. Ist eine Anwendung dergestalt auf Rechnern verteilt, kann sie durch das Anstoßen des Markenspiels im Petrinetz zur Ausführung gebracht werden.

zurück

6 Realisierung der Ausführungskomponente von VerSoS

Ziel dieser Ausführung von Funktionsnetzen ist es unter anderem, größtmögliche Nebenläufigkeit zu realisieren. Um dies zu erreichen, wird jeder terminale Knoten des Modells auf einen (Betriebssystem-) Prozeß abgebildet. Instanzprozesse bestehen dabei aus dem Baustein und einem kommunikationsfähigen Rahmen. Kanalprozesse stellen die Synchronisationspunkte dar. Sie verwalten die Daten, die zwischen den Bausteinen ausgetauscht werden. Durch die feste Zuordnung der statischen Modellinformationen auf Prozesse ist es möglich, ein relativ einfaches Protokoll zwischen den Prozessen zu installieren.

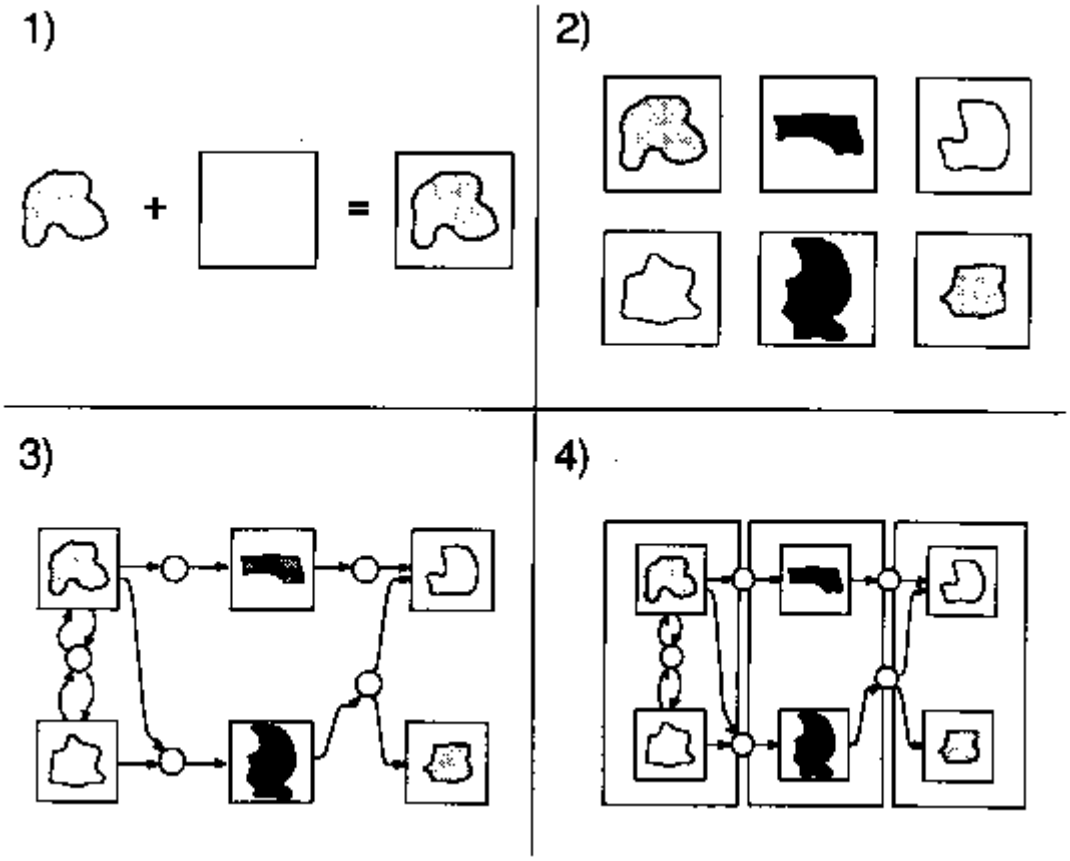

Abbildung 3: Modellierung einer verteilten Anwendung mit VerSoS

Idee der Funktionsnetze als Entwicklungsumgebung ist das Verbergen von Systemabhängigkeiten und Kommunikationsspezifika. Der Benutzer soll Bausteine und eine Spezifikation über deren Zusammengehörigkeit im System zur Verfügung stellen; die Realisierung der Ausführung wird von der Funktionsnetzumgebung übernommen. Dafür ist es notwendig, daß die Bausteine (Funktionen) eine genormte Schnittstelle besitzen. Um diese Schnittstelle wird ein Kommunikationsrahmen (siehe auch Abbildung 3, links) gelegt, der Verbindungen zu allen Kanälen unterhält, mit denen die Instanz der Spezifikation entsprechend verbunden ist. In diesem Rahmen ist der Markenspielalgorithmus der Petrinetze implementiert, also das Vorgehen, das nötig ist, um einen Baustein zu aktivieren.

Die Realisierung ist mit RPCs auf der Basis von TCP/IP-Protokollen auf SUN-Rechnern in der TFH Berlin durchgeführt worden. Es konnte gezeigt werden, daß Funktionsnetze auf die beschriebene Weise im verteilten System ausführbar sind.

Um VerSoS und die damit erstellten Anwendungen auch in heterogenen Rechnernetzen ausführen zu können, ist eine Reimplementierung mit fernen Operationen *(remote operations)* auf der Basis des ISO-OSI-Referenzmodells geplant. Die Schnittstellenbeschreibung in der Integrierten Dokumentation erfolgt dann rechnerunabhängig in der Sprache ASN.1. Die geforderte Kompatibilität zwischen Bausteinen verschiedener Implementierungen kann so gewährleistet werden.

zurück

---

7 Anwendungsbeispiel für den Einsatz von VerSoS

Wir stellen nun eine Anwendung vor, die mit VerSoS realisiert werden könnte. Gezeigt wird hier die Ausführung einer Anwendung; ihre Erstellung ist auf einer ähnlich konzipierten Oberfläche umsetzbar. Das Beispiel (das hier nur kurz skizziert werden kann) ist dem Problemkreis der Kommunikation zwischen Zulieferern und Herstellern in der Automobilindustrie entnommen [5]. Zum Austausch von elektronischen Dokumenten wird an der Schnittstelle eines Betriebes *(physical/logical interface* = P/LI) nach außen an ein Datenfernübertragungsnetz (z.B. an das nach X.25 standardisierte Datex-P) eine sogenannte CMSO-Box eingesetzt. Ihre Aufgaben sind z. B. die Bündelung bzw. Verteilung von Dokumenten, ihre Konsistenzüberprüfung mit internen Datenbeständen und eine Terminüberwachung. Die Darstellung in der Ausführung einer Anwendung ist von der zentralen Idee eines **Leitstandes** geprägt.

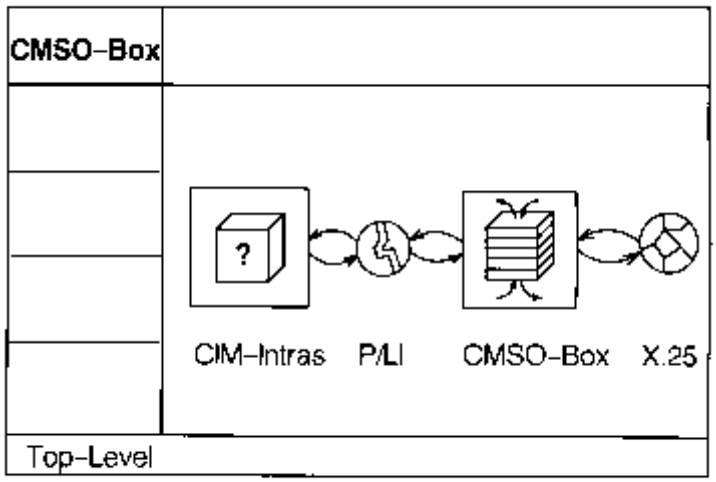

Abbildung 4: Oberfläche einer Anwendung

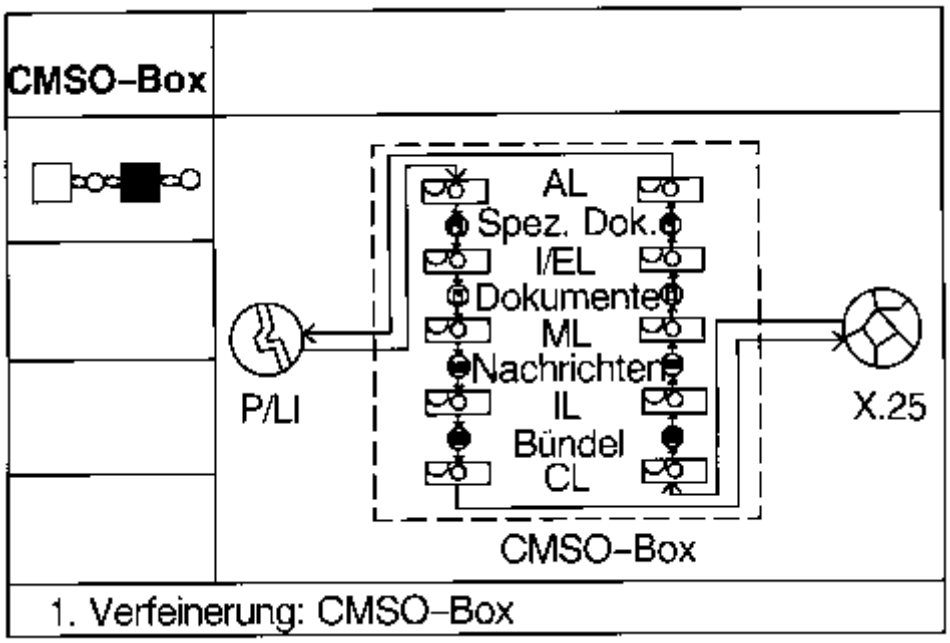

Abbildung 5: Verfeinerung in der 1. Ebene

zurück

---

8 Schlußbemerkungen

Die teilautomatisierte Erstellung von verteilten Anwendungen in einer Umgebung wie VerSoS läßt hoffen, daß die Produkte rasch entwickelbar sind und daß sie kostengünstig angeboten werden können. Die Bereitstellung einer preiswerten, neutral produzierten Software, die dabei gleichzeitig an die Anforderungen verschiedener Kunden angepaßt werden kann, fördert die Selbständigkeit und die Unabhängigkeit aller Beteiligten.

Für den Anwender, der langjährig erprobte Software weiterhin einsetzen möchte, dabei aber auf die Vorteile eines Programmverbundes im Rechnernetz nicht verzichten möchte, bietet VerSoS einen integrierten Ansatz für die Verwirklichung.

Das Konzept für VerSoS an der TFH ist erarbeitet. Einzelne Komponenten, wie z.B. in Abschnitt 6 vorgestellt, wurden bereits realisiert. Die mit Hilfe eines Netzeditors erstellten Modelle können in einem Pool von Workstations nebenläufig zur Ausführung gebracht werden. Die schrittweise Vervollständigung ist geplant.

zurück

Literatur

[l] H. P. Godbersen.

Funktionsnetze: Eine Modellierungskonzeption zur Entwurfs- und Entscheidungsunterstützung. Birkach: Ladewig-Verlag, 1983.

[2] H. P. Godbersen, K. Rastgooy, K. Schwidder.
Generierung von Programmsystemen aus graphischen Spezifikationen. Proc. 3rd Seminar on Modelling, Evaluation and Optimization of Dependable Computer Systems, Wendisch Rietz, Nov. 1990. Informatik informationen reporte, Akademie der Wissenschaften der DDR. Bd. 9/90, p. 5 - 22

[3] A. Kroll.
PIE.- Eine auf Funktionsnetzen aufbauende Software-Produktionsumgebung zur Erstellung und Ausführung verteilter Anwendungen. Berlin: Diplomarbeit an der TFH, 1991.

[4] A. Pfafferott.
Software-Synthese in verteilten Systemen. Berlin: Diplomarbeit an der TFH, 1991.

[5] H. P. Godbersen, M. Matthiesen, W. Schaber.
Modelling of Interorganisational Operations. Proc. TELEMATICS '90 Conference, BIBA/Universität Bremen, Bremen 3-5 Dec. 1990, p. 207-221

Letzte Bearbeitung:

zurück