Programm-Konstruktion
unter Verwendung loser Kopplung

Reiner W. Hartenstein

1982

Universität Kaiserslautern, Fachbereich Informatik,
Postfach 3049, D-675 Kaiserslautern 1, F. R. G.

B ü c h e r :

R. W. Hartenstein, R . Zaks : Micro-Architecture of Computer Systems; North Holland Publishing co. / American Elsevier, Amsterdam /, New York 1975

R. W. Hartenstein: Fundamentals of Structured Hardware Desi A Design Language Approach at Register Transfer Level; North Holland Publishin Co / American Elsevier, Amsterdam / New York 1977

M. Breuer, R. W. Hartenstein: Computer Hardware Description Languages and their Applications; North Holland Pu Co / American Sequoia Publishing Co; Amsterdam / New York 1981

1. EINLEITUNG

Die Technik der Datenverarbeitung leidet im Wesentlichen unter 2 Problem-Komplexen: der Software-Krise, und dem von-Neumann-Engpass. Die Software-Krise ist ein Syndrom, das letztlich vor allem auf den von-Neumann-Engpass zurueckgeht. Der Engpass ist gegeben durch den einzigen bidirektionalen Pfad, der den Zugriff zu einer CPU ermoeglicht. Dieser Pfad ist nicht nur ein Durchsatz-Engpass. Dieser Zugriffspfad bildet zusammen mit dem Dekodierer des Arbeitsspeichers einen riesigen Multiplexer, der den Arbeitsspeicher-Adressraum eindimensional definiert und somit die darin gespeicherten Daten und Befehle strukturlos anordnet. Die Aufgabe des "Software Engineering" besteht zu einem grossen Teil darin, zwecks besserer Beherrschbarkeit dem so amorph gespeicherten Code eine ordnende Struktur zu geben, wie etwa durch modulare Programmierung und Massnahmen zum Schutz von Betriebssystemen. Entsprechende Konzepte koennen jedoch vom Programmierer meist umgangen werden, da entsprechende Konzepte "soft" sind. Es wird beispielsweise immer wieder behauptet und auch bewiesen, dass das sicherste aller bekannten Betriebssysteme abstuerzen kann. Abgesehen davon verursachen darin enthaltene Praeventiv-Artefakte ("safeguards" vgl./RAM74/) zusaetzliche Kosten und Leistungsminderung. Man kann durch entsprechendes Management mit Abnahmepruefungen oder ueber zwischengeschaltete Programm-Entwicklungssysteme einen Zwang auf den Programmierer ausueben. Aber auch Management ist fehlerbehaftet und Programmierhilfen sind oft nicht verfuegbar, vor allem nicht bei Mikrorechner-Anwendungen. Strukturierungs-Massnahmen durch Aufteilung einer Anwendung in Prozesse, die dann gemeinsam an einer von-Neumann-Maschine beteiligt sind, erhoehen aber auch die Software-Komplexitaet (auch des Betriebssystemes) erheblich. Die dabei eingebrachten Strukturierungs-Artefakte sind u.a. unter den Schlagworten "context swapping overhead" /SIE75/ oder "multiplexing overhead"/HAK75/ behandelt worden.

Die Software-Krise hat zur Folge, dass in Implementierung und Betrieb die Softwarekosten die der Hardware oft um ein Vielfaches uebersteigen. Eine drastsche Erhoehung der Zahl der Betroffenen ergibt sich aus der zunehmenden Verbreitung des Mikrorechners. Die Euphorie-Phase der Mikrorechner-Anwendung geht dem Ende entgegen und eine baldige allgemeine Ernuechterung zeichnet sich ab auf Grund zunehmender Software-Erfahrung. Bei Mikrorechner-Anwendung sind die Hardware-Kosten gegenueber der Software oft sogar zu vernachlaessigen. Neben Methoden des Projektmanagement, u. U. auch mit Computer-gestuetzter Fuehrung, eignen sich auf Programmierstil und Programm-Strukturkonzepten basierende Verfahren zur Erstellung zuverlaessiger Programme. Vor Allem muessen dann zur Erreichung von Modul- und System-Robustheit folgende Arten der Kopplung zwischen Moduln angestrebt werden.

(a) schwache Kopplung
(geringer Informationsfluss zwischen den Moduln durch (2))

(b) lose Kopplung
(restriktiver Kommunikations-Mechanismus durch (1))

(c) praeventive Kopplung
(durch Eingangspruefungen an den Modul-Schnittstellen)

Waehrend (a) vor Allem eine Frage der Gestaltung und Partitionierung des Gesamtprogrammes (also des Programm-Entwurfes) ist, werden (b) und (c) durch Gestaltung der verfuegbaren Mechanismen fuer Modul-Schnittstellen stark beeinflusst. Mit diesen Mechanismen befasst sich der folgende Absatz.

2.2 DER KOPPLUNGSGRAD VON KOMMUNIKATIONS-MECHANISMEN

In /MYE76/ ist ein Klassifikations-Schema fuer geschlossene Software-Moduln angegeben, welches auf Grund der Rahmen-Spezifikation der Modul-Schnittstellen die Moduln einer von 6 Kopplungsklassen zuordnet. Ein Software-Modul laesst sich prinzipiell mit einem Knoten eines Rechnernetzes vergleichen. Der Unterschied besteht darin, dass die Kommunikation zwischen Moduln bei Netzen nicht mehr ueber das Betriebssystem, sondern ueber Hardware-Verbindungen und -Schnittstellen abgewickelt wird. Ein Knoten eines Rechnernetzes laesst sich auf Grund mehr oder weniger eingeschraenkter Zugriffs moeglichkeiten von aussen her wie ein geschlossener Unterprogramm-Modul ebenfalls einer von 6 Kopplungskategorien zuordnen (Bild 1a, nach /MYE75/). Der Rechnerknoten des Netzes ist dabei quasi ein "Hardware-gekapselter" Software-Modul.

Die sechs Kategorien erstrecken sich von der "Inhalts-Kopplung" (content coupling), der festest moeglichen Kopplung bis zur losesten Kopplung, der "Daten-Kopplung" (data coupling). Die loseste Kopplung ergibt den besten Schutz gegen Zugriffsfehler und deren Folgefehler. Die loseste Kopplung schraenkt Zugriffsmoeglichkeiten von aussen in den Modul am staerksten ein, was dem Modul damit die groesstmoegliche modulare Unabhaengigkeit verleiht. Die festeste Kopplung hingegen bietet den geringsten Schutz gegen Fehler und deren Ausbreitung in andere Moduln hinein. Die Inhalts-Kopplung (Kopplungs-Kategorie 1) liegt dann vor, wenn mehrere Prozesse (Software) oder mehrere Prozessoren auf nur einem einzigen gemeinsamen Primaerspeicher arbeiten: ein Modul kann unmittelbar zum Inhalt des anderen zugreifen. Zugriffe "hinten herum" (vgl. Einleitung) sind moeglich, unbemerkt durch den "betroffenen" Modul. Auch unkontrollierte Spruenge von einem Modul in einen anderen sind moeglich. Fast jede Aenderung in einem Modul verursacht Fehler im anderen Modul.

Bei Kategorie 2 (COMMON-Kopplung) sind nur noch die Daten global, jedoch nicht mehr der Programmcode. Die Fehleranfaelligkeit ist zwar geringer, jedoch noch immer recht hoch (vgl. /WUL73/). Die Kategorien 3 bis 5 sind in Bild 1a kommentiert. Kategorie 3 erfasst Bus-orientierte Netze, wenn das "globale Einzelfeld" der Wortlaenge eines Bus entspricht. In diese Kategorie gehoeren Netze mit beispielsweise der COMA-Schnittstelle /OVW79/. Netze mit streng lokalen "mailbox"-Speichern lassen sich in Kategorie 5 des Schema nach MYERS einordnen.

Die geringste Fehlerwahrscheinlichkeit tritt bei Daten-Kopplung auf (Kategorie 6, vgl. Bild 1a), gemaess dem MYERS'schen Schema. Hierbei sind nicht ganze Daten-Strukturen gemeinsam zugaenglich. Stattdessen koennen nur einzelne Datenobjekte uebertragen werden auf Kanaelen, die einzelnen Knotenpaaren zugeordnet sind (kuenftig "private Kanaele" genannt). Die Liste der Kopplungsklassen nach MYERS sei hier noch erweitert durch eine siebente Kategorie "Botschafts-Kopplung", bei welcher die Moduln Botschaften austauschen anstelle von Daten (Bild 1b). Daten koennen als Bestandteil von Botschaften uebermittelt werden, jedoch mit entsprechender Kennung, welche eine "Empfangspruefung" erleichtert, und somit eine weiter verbesserte Zuverlaessigkeit ermoeglicht. Eine Botschaft kann auch die Bedeutung eines Unterprogramm-Aufrufes haben. Der Unterschied zur Kopplungsklasse 4 (Steuerungs-Kopplung) besteht darin, dass der Empfaenger nicht zur Ausfuehrung der aufgerufenen Funktion gezwungen werden kann. D. h. der Empfaenger-Knoten muss eine solche Botschaft ausdruecklich akzeptieren, wenn diese ausgefuehrt werden soll.

2. ENTWURFSVERFAHREN FuER BOTSCHAFTS-GEKOPPELTE SYSTEME

Eine Methodologie fuer die Aufbereitung eines Anwendungsproblemes zwecks seiner Programmierung in ein solches Netz von Moduln hinein ist im Entstehen begriffen (s. u. a. /LEL77/, /BOC79/). Hierzu gehoert eine entsprechende Analyse des Anwendungsproblemes zur Identifikation seiner Subprozesse und der genauen

Erfassung der Beziehungen zwischen diesen Subprozessen. Nach Moeglichkeit einer Menge von Subfunktionen anzugeben (vgl. /ALF79/, /BOC79/, /FEL79/, /FIT78/, /HAZ76/, /JEN77/, /LAM78/, /LUY79/,/MES70/, /MES75/, /PAL79/, /ROS77/, /YAY79/, /ZAV 76/, /ZAV77). Bei Botschafts-Kopplung entsteht ein Netz von Erzeuger-Verbraucher-Beziehungen (vgl. Beispiel in Bild 2 a) zwischen Programm-Moduln, auch "program string" (/MOR67/, oder "verzweigte Programm-Moduln-pipeline-Structure") genannt. Eine gute Veranschaulichung dieses Aufbereitungs-problemes und der Vorgehensweise findet sich in /HRO80/.

Im Schatten der reichhaltigen Literatur ueber (in /LAU78/ "Prozedur-orientiert" genannte) Betriebssysteme mit "normalen" Kommunikations-Mechanismen gibt es einige Arbeiten ueber "Botschafts-orientierte" Kommunikation in Betriebssyst-emen. Dabei wird ein Netz von Erzeuger/Verbraucher-Beziehungen zwischen den Programm-Moduln durch das Betriebssystem realisiert (z.B. das Netz in Bild 2a durch die dem Programmierer unsichtbare Struktur im Bild 2b). Diese Arbeiten sind nicht neu und nicht ganz unbekannt (/BAL67/, /BAL71/, /BAL73/, /KNO74/, /KNU68/, /MOR67/), oder gar relativ bekannt (/BHA70/, /GOO72/, /WAL72/) und finden in juengster Zeit stark zunehmendes Interesse, vor allem wegen der einfachen Uebertragbarkeit ihrer Konzepte vom Gebiet der Betriebssysteme auf lose gekoppelte Rechnernetze. Nach /LAU78/ sind die beiden Arten von Kommunikations-Systemen zueinander dual bezueglich Durchsatz und logischer Struktur. Auch /LAU78/ bestaetigt die Aussagen des erweiterten MYERS'schen Klassifikations-Schema bezueglich modularer Robustheit: die Zustands-Information eines Prozesses benoetigt in Botschafts-orientierten Systemen keinen Schutz.

Die genannten Arbeiten ueber Botschaftssysteme liefern sehr gut brauchbare Vorschlaege fuer die Formulierung von Kommunikations-Anweisungen. Diese Vorschlaege sind Spezifikationen, geeignet etwa zur Assembler-Programmierung. Die neueren Arbeiten setzen diese Vorschlaege um in Konstrukte, die sich zur Eingliederung in hoehere Programmiersprachen eignen. Auch ADA /ADA79/ enthaelt alle notwendigen Ausdrucksmittel fuer Botschafts-Systeme. Botschafts-orientierte Betriebssysteme kommerziell nicht erhaeltlich sind, bleibt nur die Assembler-programmierte loesung, es sei denn man will sichauf das Gebiet der Betriebs-systemforschung begeben (was hier nicht der Fall ist). Das "pipe"-Konzept des UNIX-Betriebssystemes ist mit Kopplungsklasse 6 nur als eine Vorstufe zur Botschaftskopplung anzusehen. Ausserdem lassen sich hiermit keine verzweigten Kommunikations-"Netze" realisieren, da nur Kommandos der Standard-Ein/Ausgabe als Kommunikationskommandos verfuegbar sind.

3. VERBESSERTE KOMNMUNIKTIONS-SCHNITTSTELLE

Auf Grund der Erfahrungen des ESRA-Projektes wurde aus dessen Kommunikations-Schnittstelle eine verbesserte Version entwickelt, die sich noch besser fuer eine Foerderung robuste Modularitaet lose gekoppelter Programme eignet. Zwecks Erzielung besserer Modul-Robustheit, geringerer Blockierungs-Anfaelligkeit und besserer Diagnostizierbarkeit von sogenannten "Rythmus-Stoerungen" (die ja typisch sind fuer Botschafts-gekoppelte Systeme), wurde folgender neuer Kopmmunikationsmechanismus entwickelt. In Anlehnung an das THOTH-Betriebsszstem /GENT 81/ wurden die folgenden grundlegenden Einschraenkungen eingefuehrt:

- der Botschaftspuffer hat nur die Laenge 1 (die Entstehung von Warteschlangen ist damit ausgeschlossen)

- die Botschaften haben einheitliches Format (dem Benutzer ist anheim gestellt, dieses evtl. nicht voll zunutzen)

Unter diesen Randbedingeungen sind die folgenden Kommunikationsanweisungen realisiert:

```
SEND ( Verbraucher, Erzeuger, Botschaft, Status, time-out ) ;

UNSEND ( Verbraucher, Erzeuger ) ;
```

Eine SEND-Anweisung bietet dem Verbraucher eine Botschaft an und wartet auf dessen Antwort durch RECEIVE (ohne Rueckbotschaft) oder REPLY (mit Rueckbotschaft). Eine SEND-Operation wird bei Ausbleiben einer Antwort entweder nach Erreichen des "time-out" automatisch abgebrochen, oder durch eine UNSEND-Anweisung des Sendermoduls rueckgaengig gemacht. Weitere Kommunikations-Kommandos sind:

```
REPLY ( Erzeuger, Verbraucher, Botschaft, Status, time-out ) ;

UNREPLY ( Erzeuger, Verbraucher ) ;

RECEIVE ( Erzeuger,  Verbraucher, Status, time-out ) ;

UNRECEIVE ( Erzeuger,  Verbraucher ) ;
```

Eine RECEIVE- oder REPLY-Anweisung empfaengt eine bereits vorliegende Botschaft oder wartet auf eine noch nicht vorliegende Botschaft. Der Abbruch einer solchen Anweisung erfolgt entweder automatisch nach dem Verstreichen der durch den "time-out"-Parameter angegebenen Zeit, oder programmiert durch eine UNREPLY- bzw. eine UNRECEIVE-Anweisung. Das REPLY sendet eine Antwort-Botschaft zurueck an den sendenden Modul. waehrent das RECEIVE nur zu Synchronisationszwecken ein Quittungs-Signal zuruecksendet.

3.1 PROGRAMMKONSTRUKTION AUF DER BASIS LOSER MODUL-KOPPLUNG

Wenn ausschliesslich die o. g. nannten Kommandos zur Kommunikation zwischen verschiedenen Moduln eines Programmes zur Verfuegung stehen, ergeben sich entscheidende Aenderungen des Programmierstiles und der Software-Zuverlaessigkeit. Dies wurde durch mehrere andere Papiere aus dem ESRA-Projekt ausfuehrlich begruendet. Besonders "revolutionierend" wirkt dies auf den Programm-Entwurf, also auf die Gesamt-Partitionierung des Gesamt-Programmes. Die Unmoeglichkeit der Vereinbarung globaler oder gemeinschaftlicher Datenstrukturen erzwingt solche Partitionierungen, die entweder eine geringe Kommunikationsrate oder die Uebertragung sogenannter "token strings" (Folgen kleinerer geschlossener Datenobjekte) zum Daten-Transfer zwischen Moduln verlangt.

Die Konsequenzen sind die folgenden: einerseits wird eine robustere Modularitaet erzwungen. Andererseits ist eine Umstellung des Stils des Programm-Entwurfes notwendig, denn die Partitionierung muss auf eine Weise erfolgen, die grundlegend verschieden ist von der gewohnten Methode unter Zulaessigkeit deklarierter globaler Datenstrukturen. Hierzu soll durch eine Reihe von Diplom- und Studienarbeiten auf der verfuegbaren ESRA-Hardware Erfahrung gewonnen werden.

Interessant ist hierbei, dass auch die Partitionierung von Hardware aus integrierten Schaltungen aehnliche Zwaenge, wie sie ogen angegeben wurden,

hat: Die Verarbeitungsrate der Chip-Peripherie ist betraechtlich geringer als die des Chip-Inneren, denn die Zahl der Anschlussfahnen ist beschraenkt (meist auf weniger als 100) und die Schaltgeschwindigkeit ist geringer (eine Groessenordnung oder mehr). Wir sehen also eine Verwandschaft des Entwurfs-Stiles zwischen lose gekoppelter Software und aus mehreren "Chips" bestehender Hardware. Integriertes "Hardware/Software Engineering" und auch der Ersatz von Software-Moduln durch Hardware werden hierdurch erleichtert.

Da bei Verwendung botschafts-orientierter Betriebssysteme der Benuetzer nicht die zentrale Vermittlungsfunktion des Betriebssystems (vgl. Beispiel in Bild 2b), sondern nur das Netz der Kommunikationskanaele sieht (vgl. Bild 2a im Gegesatz zu Bild 2b), ist der Ersatz des Betriebssystemes durch ein Netz von Hardware-Kanaelen kein Problem. Die Kommunikations-Anweisungen bleiben unveraendert. Die verfuegbare ESRA-Hardware ist eine gut geeignete Alternative fuer ein Botschafts-orientiertes Betriebssystem, welcher uns derzeit nicht zur Verfuegung steht. Beim ESRA-Netzbaukastens wurden die Kommunikationsmechanismen in 8080-Assembler implementiert in Verbindung mit einer speziellen Hardware-Schnittstelle und steckbaren privaten Kanaelen. Eine Reihe von Publikationen beschreiben Einzelheiten hierzu.

3.2 ROBUSTE SOFTWARE-BAUKAESTEN ?

Die durch die sehr lose Kopplung (Kategorie Nr. 7) gegebene hohe modulare Unabhaengigkeit der Software einzelner Knoten foerdert deren Verwendung als vorgefertigte Knoten "ab Lager" mit "canned software", die quasi "hardwaregekapselt" und somit gut geschuetzt im Knoten untergebracht ist. Die Notwendigkeit, solche Knoten ab Lager evtl. an ein spezielles Anwenderproblem anpassen zu muessen, ist entweder ueberhaupt nicht oder nur selten gegeben, wenn die Knoten-Spezifikation gut ueberlegt, ausgereift und erprobt ist. Ein solcher "Knoten ab Lager" ist wegen seines eigenen Rechners autonom und kann somit relativ leicht durch hardwaremaessigen Anschluss an ein Testgeraet geprueft werden. Die Schaffung eines flexiblen Bausteinprogrammes fuer ein Spektrum von Problemen aus einer bestimmten Klasse von Anwendungen wird hierdurch erheblich erleichtert. Ein solches Buausteinprogramm wird ein Repertoir an speziellen internen Knoten und Randknoten umfassen derart, dass etwa ein einfaches System durch sukzessives Hinzufuegen weiterer Knoten (ab Lager) schrittweise erweitert werden und in seiner Leistungsfaehigkeit gesteigert werden kann. Auf diese Weise laesst sich ein Modul-System fuer eine vielseitige Produktpalette aufruestbarer Systeme entwickeln.

4. LITERATURHINWEISE

/ADA79/ N. N., "Preliminary ADA Reference Manual", SIGPLAN Notices 14, No. 6/7 (June/July 1979).

/ALF79/ Alford, m., "Requirements for Distributed Data Processing Design", Proc. 1st Int'l. Conf. on Distr. Computing Systems, Huntsville, Oct. 1979.

/ANJ75/ Anderson,G.A., Jensen, E.D., "Computer Interconnection Structures, Taxonomy, Characteristics, and Examples", Computing Surveys, vol.7, (Dec.1975).

/AVI78/ Avizienis, A., "Fault-tolerance: The Survival Attribute of Digital Systems", Proc. IEEE 66 (1978), No. 10 (Oct.).

/BAL67/ Balzer, R. M., "Inter-Program Communication and Job Control" Proc. AFIPS 1967 SJCC.

/BAL71/ Balzer, R. M., "Ports - A Method for Dynamic Interprogram Communication and Job Control", R-605 ARPA, Rand Corp., Santa Monica, CA, 1971.

/BAL73/ Balzer, R. M., "An Overview of the ISPL Computer System Design", Comm. ACM 16 (2), 1973.

/BAL76/ Ball, J. E., Feldman, J. R., Low, J. R., Rashid, R. F., Royner, P. D., "RIG - Rochesters Intelligent Gateway", IEEE-Trans. SE-2, No. 4 (Dec. 1976).

/BAL79/ Ballard, D. R., "Designing fail-safe microprocessor systems", Electronics, Jan. 4, 1980.

/BAL80/ Ball, J. R., Burke, E. J., Gertner, I., Lantz, K. E., Rashid, R. F., "Perspectives on Message-based Distributed Computing", Proc. Computer Networking Symp., Gaithersburgh, VA, 1980

/BHA70/ Brinch Hansen, P., "The Nucelus of a Multiprogramming-System", Comm. ACM 13, No. 4 (April 1970).

/BHS79/ Bellm, A., Hoerbst, E., Sanderweg, G., Sauer, A., Scheiterer, E., Schmidtke, F., Thinschmidt, H., "Ein Multi-Mikrocomputer am Arbeitsplatz", Elektronik Bd. 28 (1979), H. 19 bis 23.

/BIT75/ Bisiani, R., Tisato, F., "A multi-microprocessor system as a set of cooperating processes", in: /HAZ75/.

/BOC79/ Bochmann, G. v., "Architecture of Distributed Computer Systems", Lecture Notes in Computer Science No. 77, Springer-Verlag, Heidelberg/New York, 1979

/BUR80/ Burke, E. J., Rashid, R. F., "A Capability-based Message Passing System", Technical Report, Computer Science Department, University of Rochester, Rochester, NY, 1980.

/CAS76/ Casaglia, G.F., "Distributed Computing Systems: a biased review", EUROMICRO Newsletter, no.2 (Oct. 1976)

/DES78/ Despain, A. M., Patterson, D. A., "The Computer as a Component", Computer Science Division, University of California, Berkeley, CA, 1978

/DIE79/ Dieckmann, J., "Entwurf spezialisierter Rechnernetze zur Unterstuetzung modularer Programmierung", int. Bericht 9/79, FB. Informatik, Universitaet Kaiserslautern, Juni 1979.

/DIH80/ Dieckmann, J., Hartenstein, R. W., "A local Micro Network to support Software Modularity", in: /SAH80/.

/ENG74/ England, D. M., "Capability Concept Mechanism and Structure in SYSTEM 250", Proc. Int'l. Workshop on Protection in Operating Systems, IRIA, Roquencourt, France, Aug. 1974.

/ENS74/ Enslow, Ph.H., "Multiprocessors and parallel processing", John Wiley, New York 1974

/FAG78/ Faggin, F. "How VLSI impacts computer architecture" IEEE Spectrum, vol. 15, (May 1978)

/FEL77/ Feldmann, J. A., "Programming Methodology for Distributed Computing", TR9, Dept. of CS, University of Rochester, 1977

/FEU73/ Feustel, E. A., "On the Advantages of Tagged Architectures", IEEE Trans. C-22, No. 7 (July 1973).

/FIT78/ Fitzwater, D. R., "A Decomposition of the Complexity of System Development Process", COMPSAC '78 Proceedings.

/FUL78/ Fuller, S.H., "Multi-Microprocessors an Overview and Working-Example", Proc. IEEE, vol. 66 (Febr. 1978)

/GEN81/ Gentleman, W. M., "Message passing between sequential processes: the reply primitive and the administrator concept", Software Practive and Experience, Vol. 11 (1981), p. 435 - 466,

/GIL80/ Giloi, W. K., "Rechnerarchitektur", Informatik-Spektrum 3 (1980).

/GOO72/ Goos, G., "The Operating System BSM", internal report, Math. Dept., Techn. University Munich, 1972.

/HAK75/ Hartenstein, R.W., Koch, G., "The Universal Bus Considered Harmful", in: /HAZ75/

/HAM72/ Hamer-Hodges, K. J., "Fault Resistance and Recovery within SYSTEM 250", Proc. 1st Int'l Conf. on Computer Comm., IEEE New York 1972

/HAR77/ Hartmann, A., "A Concurrent Pascal Compiler for Minicomputers", Lecture Notes in Computer Science, Vol. 50, Springer Verlag 1977

/HAZ75/ Hartenstein, R.W., Zaks, R., "Microarchitecture of Computer Systems", North Holland Publ., Co./American Elsevier, Amsterdam/New York 1975

/HAZ76/ Hamilton, M., Zeldin, S., "Higher Order Software - A Methodology for Defining Software", IEEE Trans. SE-2, No. 1 (Mrch. 1976).

/HHP79/ Hartenstein, R.W., Hoerbst, E., von Puttkamer, E., "Loosely coupled micros - distributed function architecture", in: /TCL79/

/HRO80/ Haendler, W., Rohrer, H., "Gedanken zu einem Rechner-Baukasten-System", elektronische Rechenanlagen 22 (1980), H. 1.

/HTN77/ Hartenstein, R. "Fundamentals of Structured Hardware Design" North Holland/American Elsevier, Amsterdam/New York 1977.

/JEN77/ Jenny, G. J., "Process Partitioning in Distributed Systems", Proc. Nat. Telecomm. Conf., Los Angeles, Cal., 1977.

/KAY74/ Kane, J. R., Yau, S. S., "Concurrent Software Fault Detection", IEEE T-SE-1, No. 1 (Mrch 1974).

/KNO74/ Knott, G. D., "A Proposal for Certain Process Management and Intercommunication Primitives", Operating Systems Review 8, (4), 1974.

/KNU68/ Knuth, D. E., "The Art of Computer Programming, Vol. 1: Fundamental Algorithms", Addison Wesley, Reading, Mass., 1968.

/KON79a/Konrad, W., "Asynchroner Datenpfad zur losen Kopplung von Mikrorechnern", Interner Bericht Nr. 13/79 am Fachbereich Informatik der Universitaet Kaiserslautern, 1979

/KON79b/Konrad, W., Communication and Testing in A Loosely Coupled Multi Microcomputer System", Preprints ACM-Workshop on Microcomputers, Munich, F.R.G., Oct. 1979

/LAM76/ Lamb, S. S. et al., "SAMM - A Modelling Tool for Requirements and Design Specificatikon", COMPSAC '78 Proceedings.

/LAU78/ Lauer, H. C., Needham, R. M., "On the Duality of Operating System Structures", Proc. 2nd Int'l. Symposium on Operating Systems, IRIA, Roquencourt, France, 1978.

/LEL77/ Le Lann, G., "Distributed Systems - Towards a Formal Approach", Proc. IFIP-Congress Stockholm 1977.

/LIS74/ Liskov, B. H., Zilles, S. "Programming with abstract Data Types", SIGPLAN Notices 9 (1974), April.

/LIS77/ Liskov, B. H., "Abstraction Mechanisms in CLU", Comm.ACM 20 (1977), 8 (August).

/LIS79/ Liskov, B. H., "Primitives for Distributed Comnputing", Asilomar worksh. on Operating Syst. Principles, ACM, New York 1979

/LUY79/ Lu, P. M., Yau, S. S., "A Methodology for Representing Formal Specifications for Distributed Computing System Software Design", Proc. 1st Int'l. Conf. on Distr. Computing Systems, Huntsville, Oct. 1979.

/MCQ78/ MacQueen, D. B., "Models for Distributed Computing", Proc. EEC/ IRIA Course on Distr. Processing, Nice, France, July 1978

/MEL77/ Mellor, F., Olden, W. J., Bedard, C. J., "A Message-switched Operating System for a Multiprocessor", Proc. COMPSAC 1977, IEEE Computer Society, New Zork, 1977.

/MES70/ Mesarovic, M. D. et al., "Theory of Hierarchical Systems", Academic Press, 1970.

/MET75/ Mesarovic, M. D., Takahara, Y., "General System Theory: Mathematical Foundations", Academic Press, 1975.

/MOR67/ Morenoff, E., McLean, J. B., "Inter-program Communications, Program Strings, and Buffer Files", AFIPS 1967 SJCC.

/MYE75/ Myers, G. J., "Reliable Software through Composite Design", Petrocelli & Carter, New York, 1975.

/MYE76/ Myers, G. J., "Software Reliability - Principles and Practices", John Wiley & Sons, New York, 1976.

/MYE78/ Myers, G.J., "Advances in computer architecture", John Wiley & Sons, New York, 1978.

/NEH79/ Nehmer, J., "The Implementation of Concurrency for a PL/I-like Language", Softw. Pract. and Experiences 9 (1979), December.

/NEH80/ Nehmer, J., "Implementierungs-Techniken fuer Monitore", interner Bericht 1/80, Fachber. Informatik, Univ. Kaiserslautern, 1980.

/NIL80/ Nilsson, S. A., "M3R - ein modulares Mehrmikrorechner-System mit Prozesssicherungsstruktur", elektronische Rechenanlagen 20 (1978), H. 3.

/OVW79/ van der Ouderaa, E.M.A.M., Vrielink, H., Willemse, A., "COMA, a single-chip communication interface for distributed microcomputer Systems" in: /TCL79/

/PAI79/ Painke, H., "Der Einfluss von VLSI auf die Systemstruktur", NTG-Fachtagung "Hoechstintegrierte Schaltungen, Technologie, Schaltungstechnik, Systeme, Anwendungen", Baden-Baden 1979, NTG-Fachberichte, Bd. 68, VDE-Verlag, Berlin, 1979

/PAL79/ Palmer, D. F., "Distributed Computing System Design at the Subsystem/Network Level", Proc. 1st. Int`l. Conf. on Distr. Computing Systems, Huntsville, 1979.

/RAM74/ Ramamoorthy, C.V., Cheung, R.C., Kim, K.H., "Reliability and Integrity of Large Computer Programs", Memo No. ERL-M430 University of California, Berkeley, CA, 1974

/RAM78/ Ramamoorthy, C.V. So.H.H., "Software Requirements and Specification", in: /RAY78/.

/RAY78/ Ramamoorthy, C.V., Yeh, R.T., "Tutorial: Software Methodology", IEEE Computer Society, Long Beach, Ca, 1978

/RAN78/ Randell, B., et al., "Reliability Issues in Computing Systems Design", Computing Surveys 10, No. 2 (June 1978).

/ROS77/ Ross, D., "Structured Analysis (SA)", IEEE Trans. SE-3, No.1 (Jan. 1977).

/SAH80/ Sami, M. G., Hanna, K., Thompson, L., "Proceedings 1980 EURMIOCRO Symposium, London, Sept. 1980", North Holland Publishing Co., Amsterdam /New York 1980.

/SIE75/ Siewiorek, D.P., "Process Coordination in Multiprocessor Systems", in: /HAZ75/

/SUM77/ Sutherland, I. E., Mead, C. A., "Microelectronics and Computer Science", Scientific American 273, No. 3 (Sept. 1977).

/TCL79/ Tiberghien J., Carlstedt, G., Lewi, J., (eds), "Microprocessors and their applications", North Holland Publ.Co., Amsterdam/ New York/Oxford 1979

/THF79/ Thurber, K. J., Freeman, H. A., "Architecture Considerations for Local Computer Networks", Proc. 1st. Int'l. Conf. on Distrib. Computing Systems, Huntsville, Oct. 1979.

/WAL72/ Walden, N., N., "A System for Subprocess Communication in a Resource-Sharing Computer Network", Comm. ACM 15 (1972).

/WUS73/ Wulf, W., Shaw, M., "Global variable considered harmful", SIGPLAN Notices, Vol. 8, No. 2 (Feb. 1973)

/YAC75/ Yau, S.S., Cheung, R. C., "Design of Self-Checking Software", SIGPLAN Notices 10, No. 6 (June 1975), (Proc. Int'l Conf. on Reliable Software, Los Angeles, CA.).

/YAY79/ Yau, S. S., Yang, C. C., "An Approach to Distributed Computing System Software Design", Proc. 1st. Int'l. Conf. on Distrib. Computing Systems, Huntsville, Oct. 1979.

/ZAV76/ Zave, P., "On the Formal Definition of Processes", Conf. on Parallel Processing, Wayne State Univ., IEEE Computer Society 1976.

/ZAV77/ Zave,P., "A Design Tool for Real-Time Processes", Conf. on Information Sciences and Systems, The Johns Hopkins Univ., Philadelphia, 1977.

/ZIL73/ Zilles, S. N., "Procedural encapsulation: a linguistic protection technique", SIGPLAN Notices 8, no. 9, (Sept. 1973).

Kopplungs-Kategorie	Kopplungs-Grad	Modul-Attribute	Kommentar
Inhalts-Kopplung	1 (festeste Kopplung)	ein Modul greift uneingeschraenkt in einen anderen Modul	Aenderungen in einem Modul verursachen mit hoher Wahrscheinlichkeit Fehler in einem anderen Modul - illegale schwer und spaet zuentdecken
COMMON-Kopplung	2	wahlfreier direkter Zugriff zu gemeinsamen globalen Datenbereichen, Speichern oder Rwgistern	unkontrollierbare Zugriffe: unbemerkt durch andere Moduln, daher kein Schutz und somit geringe Zuverlaessigkeit
Extern-Kopplung	3	direkter Zugriff zu globalem Ein-Feld-Datenobjekt (z.B. EXTERNAL-deklarierte Variable wie zw. PL/I-Moduln)	Probleme aehnlich wie bei COMMON-Kopplung, jedoch ohne Probleme bezuegl. der physichen Anordnung der Daten-Objekte
Steuerungs-Kopplung (control cplg.)	4	Ein Modul steuert explizit die Funktion eines anderen Moduls (z.B. durch Funktionscode)	Der aufrufende Modul benoetigt Kenntnis der Logik des aufgerufenen Moduls: daher eingeschraenkte Unabhaengigkeit
Struktur-Kopplung ("stamp" coupling)	5	Moduln greifen paarweise wahlfrei zu nicht-globaler Datenstruktur zu (z.B.: nicht-globaler gemeinsamer File, od.lokale "mail box"	unnoetig feste Kopplung ueber vollstaendige Records anstelle der daraus benoetigten einzelnen Datenobjekte: Format-Empfindlichkeit
Daten-Kopplung	6 (loseste Kopplung n. MYERS)	nur Einzel-Datenobjekte werden uebergeben (keine Datenstrukturen), die direkt verlangt sind	nur minimnale Informations-Mengen werden uebertragen, daher geringere Fehlerwahrscheinlichkeit als bei 5
a)			
Botschafts-Kopplung (message cplg.)	7	Einzelobjekte werden zusammen mit Objekt-Kennung uebergeben	wie bei 6, jedoch mit Pruefung vor Uebernahme, daher zusaetzlicher Schutz gegen Fehler (Kopplung noch loser als bei 6 wegen zwischengeschaltetem Schutzmechanismus)
b)			

Bild 1. Kategorien der Kopplung zwischen Moduln: a) nach MYERS, b) Erweiterung.

Kopplungs-Kategorie	Kopplungs-Grad	Modul-Attribute	Kommentar
Inhalts-Kopplung	1 (festeste Kopplung)	ein Modul greift uneingeschraenkt in einen anderen Modul	Aenderungen in einem Modul verursachen mit hoher Wahrscheinlichkeit Fehler in einem anderen Modul - illegale schwer und spaet zuentdecken
COMMON-Kopplung	2	wahlfreier direkter Zugriff zu gemeinsamen globalen Datenbereichen, Speichern oder Registern	unkontrollierbare Zugriffe: unbemerkt durch andere Moduln, daher kein Schutz und somit geringe Zuverlaessigkeit
Extern-Kopplung	3	direkter Zugriff zu globalem Ein-Feld-Datenobjekt (z.B. EXTERNAL-deklarierte Variable wie zw. PL/I-Moduln)	Probleme aehnlich wie bei COMMON-Kopplung, jedoch ohne Probleme bezuegl. der physichen Anordnung der Daten-Objekte
Steuerungs-Kopplung (control cplg.)	4	Ein Modul steuert explizit die Funktion eines anderen Moduls (z.B. durch Funktionscode)	Der aufrufende Modul benoetigt Kenntnis der Logik des aufgerufenen Moduls: daher eingeschraenkte Unabhaengigkeit
Struktur-Kopplung ("stamp" coupling)	5	Moduln greifen paarweise wahlfrei zu nicht-globaler Datenstruktur zu (z.B.: nicht-globaler gemeinsamer File, od.lokale "mail box"	unnoetig feste Kopplung ueber vollstaendige Records anstelle der daraus benoetigten einzelnen Datenobjekte: Format-Empfindlichkeit
Daten-Kopplung	6 (loseste Kopplung n. MYERS)	nur Einzel-Datenobjekte werden uebergeben (keine Datenstrukturen), die direkt verlangt sind	nur minimale Informations-Mengen werden uebertragen, daher geringere Fehlerwahrscheinlichkeit als bei 5
a)			
Botschafts-Kopplung (message cplg.)	7	Einzelobjekte werden zusammen mit Objekt-Kennung uebergeben	wie bei 6, jedoch mit Pruefung vor Uebernahme, daher zusaetzlicher Schutz gegen Fehler (Kopplung noch loser als bei 6 wegen zwischengeschaltetem Schutzmechanismus)
b)			

Bild 1. Kategorien der Kopplung zwischen Moduln: a) nach MYERS, b) Erweiterung.

Schriftenverzeichnis zum Arbeitsgebiet "Mikrorechnernetze feiner Granularität"

1. Werner Konrad, "Asynchroner Datenpfad zur losen Kopplung von Mikrorechnern", Universität Kaiserslautern, 1979.

2. Arwin Stuber, "Bereitstellung zweier Hardware-Komponenten zur Implementierung eines zentralen Test- und Überwachungssystem für lose gekoppelte Mikrorechner", Studienarbeit, Universität Kaiserslautern,

3. R. W. Hartenstein, E. Hörbst, E. von Puttkamer, "Loosely coupled micros - distributed function architectures: a design kit and development tool", Universität Kaiserslautern, Fachbereich Informatik, Interner Bericht 13/79.

4. J. Dieckmann, R. W. Hartenstein, "A local micro network to support software modularity", University of Kaiserslautern, Interner Bericht 26/80.

5. J. Dieckmann, R. W. Hartenstein, Universität Kaiserslautern, "A local micro network to support software modularity", Short version, presented at the 6th Int'l. EUROMICRO Symposium, London, England, September 1980.

6. J. Dieckmann, M. Flötotto, R. Hartenstein, W. Konrad, F. B. Informatik, H. Bellm, A. Sauer, E. Schmitter, "Die Realisierung des ESRA-Rechnernetz-Baukasten-Systems: ein Ansatz zu einer Methodologie für zuverlässige Software", interner Bericht, Universität Kaiserslautern, Mai 1980.

7. R. W. Hartenstein, W. Konrad, Computer Science Dept., A. Sauer, Munich, "A loosely coupled multi-microstructure as a tool for software development", Interner Bericht, Universität Kaiserslautern, 1980.

8. J. Dieckmann, R. W. Hartenstein, "The ESRA Micro Network Kit for Hardware-Aided Support of Software Reliability", ICCC International Conference on Circuits and Computers, New York, USA, October 1980.

9. J. Dieckmann, "Entwurf spezialisierter Rechnernetze zur Unterstützung modularer Programmierung", Universität Kaiserslautern, Interner Bericht 9/79.

10. Werner Konrad, "Communication and Testing in a Loosely coupled Multi-Microcomputer System", ACM-work-shop on microcomputing, München, F.R.G. , 24/25, Oktober 1979.

11. M.Flötotto; R.W.Hartenstein: "Run-time diagnostics of faults and communication indigestion in a loosely coupled local microcomputer network";
Proceedings of the workshop on Self Diagnosis and Fault-Tolerance; Universität Tübingen, Juli 1981.

12. R. Hartenstein: Abschluß-Bericht des ESRA-Projekt; Kaiserslautern, 1981 (2. Auflage: April 1984)