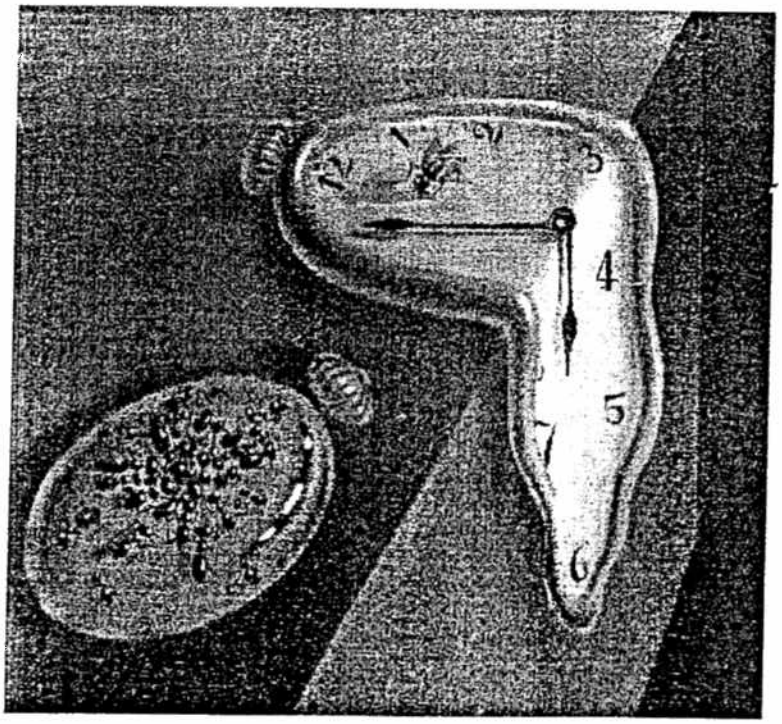
Institut für Automation
Abt. für Automatisierungssysteme

Technische
Universität
Wien

Projektbericht Nr. 183/1-22
Juni 1991

FWF-Projektantrag Versatile Timing Analyzer (VTA)

U. Schmid



Ausschnitt aus: Salvador Dali, "Die Beständigkeit der Erinnerung"

Projektantrag VTA

Ulrich Schmid, 9. 11. 1990

Kurzbeschreibung Projekt VTA

Das Projekt VTA (*Versatile Timing Analyzer*) hat die Entwicklung geeigneter Meßverfahren bzw. letztlich das Design eines Prototyps für ein System zur Erfassung des Zeitverhaltens von verteilten Echtzeitsystemen zum Inhalt. Unter einem *Echtzeitsystem* ist hier ein spezielles Computersystem zur Kontrolle eines zeitkritischen *technischen Prozesses* zu verstehen; Beispiele dafür wären etwa Automatisierungssysteme für Fertigungsstraßen, Kraftwerke, Flugzeuge und ähnliches. Es ist offensichtlich, daß bei derartigen Systemen zusätzlich zu der ohnedies selbstverständlichen *Korrektheit* von Verarbeitungsergebnissen auch deren *rechtzeitiges Vorliegen* garantiert werden muß; ein richtiger Stellbefehl an ein Kühlventil, eine Sekunde zu spät geliefert, kann unter Umständen dieselben Folgen haben wie ein falscher (oder gar keiner). Einem allgemeinen Trend in der Informatik Rechnung tragend, haben auch im Bereich der Echtzeitsysteme die *verteilten* oder *parallelverarbeitenden Systeme* Einzug gehalten. Die einem einzelnen (Groß-)Computer unter anderem in Hinblick auf Zuverlässigkeit und Preis/Leistungsverhältnis überlegenen verteilten Systeme bestehen aus mehreren, über verschiedenste Kommunikationsmedien (etwa Local Area Networks, LANs) gekoppelten Rechnern.

Obwohl durch den Einsatz parallelverarbeitender Systeme eine bedeutende Steigerung der Ausführungsgeschwindigkeit möglich ist, löst dies nicht das Problem der Rechtzeitigkeit von Verarbeitungsergebnissen. Komplexe zeitliche Abhängigkeiten von Verarbeitungsschritten untereinander bringen es mit sich, daß trotz immer größer werdender (Gesamt-)Rechenleistung gewisse Zeitschranken nicht unterschritten werden können. Diese Tatsache führt auch dazu, daß das Problem der Rechtzeitigkeit in der Praxis nur unvollständig in den Griff zu bekommen ist. Es ist z.B. in den heutzutage sehr häufig verwendeten *ereignisgesteuerten Echtzeitsystemen* schon einmal kaum möglich, alle möglichen Abfolgen einer parallelen Verarbeitung zu betrachten, da deren Anzahl jeden diesbezüglichen Versuch hoffnungslos macht. Derartige Schwierigkeiten führen dazu, daß der ganze Problemkreis in der einschlägigen Industrie oft schlichtweg ignoriert oder mit untauglichen Mitteln behandelt wird (etwa nach dem Motto: wenn ein Echtzeitsystem eine Woche praktischen Betrieb ohne Absturz "überlebt", dann wird es wohl auch weiterhin rechtzeitige Verarbeitungsergebnisse liefern ... . Unter anderem am Beispiel des Atomkraftwerkes *Harrisburg* hat sich jedoch gezeigt, daß derartige Hoffnungen nur bis zu dem zufälligen Zusammentreffen gewisser Umstände berechtigt sind!)

Der im Rahmen des beantragten Projekts zu entwickelnde VTA stellt nun ein Werkzeug dar, mit dem diesem Problemkreis etwas effektiver zu Leibe gerückt werden kann. Es handelt sich bei diesem VTA um ein System, welches kontinuierlich, während des Betriebs eines verteilten Echtzeitsystems, beliebig viele, frei definierbare Zeitmessungen vornimmt. Auf diese Weise kann z.B. die Zeit zwischen der Erfassung des Meßwerts von einem Temperaturfühler bis zur Ausgabe des Stellsignales an ein Kühlventil kontinuierlich gemessen und damit festgestellt werden, ob genügend "Zeitreserve" vorhanden ist. In diesem Zusammenhang stellen sich konkrete wissenschaftliche Fragen. So muß etwa eine Methode zur Erfassung frei definierbarer Ereignisse (Events) gefunden werden, die den Normalbetrieb des Echtzeitsystems nicht beeinflußt. Damit verbunden ist auch eine Untersuchung darüber, wie in einem verteilten System eine gemeinsame "Zeitbasis" geschaffen werden kann, die die richtige Ordnung der Ereignisse, die ja auf verschiedenen Rechnern eintreten können, gewährleistet. Neben der Erfassung stellt auch die Datenreduktion (Abstraktion) einen wichtigen Problemkreis dar, da die riesige Menge anfallender Information nicht ohne speziell zu entwickelnden Methoden verarbeitet werden kann. Letztlich stellt sich noch die Frage, auf welche Art und Weise die gesammelte Information statistisch ausgewertet vor allem graphisch repräsentiert werden kann. Die Antworten auf diese Fragen sind übrigens nicht nur im Zusammenhang mit dem VTA relevant, sondern von allgemeiner Bedeutung. So können sie nach entsprechender Überarbeitung auch für das im Planungsstadium befindliche Projekt

Real Time Simulation verwendet werden.

Abseits von dem für die Praxis so wichtigen Einsatzgebiet des VTAs für das *Monitoring von Echtzeitsystemen* stellt dieser auch ein ganz wesentliches Hilfsmittel dar, um dem Problem *"Performance–Requirements für Echtzeitsysteme"* praktisch nähertreten zu können. Dabei geht es grob um die Bereitstellung von (theoretischen) Modellen, die letztlich (in einer frühen Phase der Entwicklung) Aussagen über die nötige Rechnerleistung ermöglichen; "klassische" Performance–Maße wie die mittlere Auslastung usw. sind in Hinblick auf Zeitbedingungen leider relativ bedeutungslos.

Abschließend ist zu bemerken, daß in der internationalen Forschung kaum Aktivitäten in dieser Richtung gefunden werden konnten. Dies gibt zu der Hoffnung Anlaß, daß im Falle der Förderung des Projektes im geplanten Umfang eine österreichische Forschungsaktivität nicht (wie so oft) im Schatten der Forschungstätigkeiten anderer Länder, sondern an der Spitze stehen könnte.

1. Problemstellung/Stand der Forschung

Gegenstand des vorliegenden Projektes *VTA* (*Versatile Timing Analyzer*) ist die Entwicklung geeigneter Meßverfahren bzw. letztlich eines geeigneten Tools zur Erfassung des Zeitverhaltens von verteilten Echtzeitsystemen.

1.1 Allgemeines

Unter einem *Echtzeitsystem* ist im folgenden ein *Computersystem* zur Kontrolle eines (zeitkritischen) *technischen Prozesses* zu verstehen; Beispiele dafür wären etwa Automatisierungssysteme für Fertigungsstraßen, Kraftwerke, Flugzeuge und ähnliches. Mit dem Terminus *Echtzeit-Software* wird natürlich eine für Echtzeitsysteme geeignete Software bezeichnet. Im Unterschied zu konventioneller Software ist Echtzeit-Software in der Regel

- **Ereignisorientiert**
  Die Notwendigkeit für Aktivitäten der Software (also "Berechnungen") ergibt sich im allgemeinen aufgrund einer Vielzahl verschiedener externer Ereignisse (= Zustandsänderungen im technischen Prozeß), die meist unvorhersagbar und (scheinbar) unkorreliert eintreten. Bei einem typischen System für die Industrieautomatisierung betrifft dies z.B. die aus dem technischen Prozeß kommenden Signale von diversen Sensoren (Temperaturfühler, Lichtschranken, ...).

- **Zeitkritisch**
  Zusätzlich zu der ohnedies selbstverständlichen *Korrektheit* von Verarbeitungsergebnissen ist es auch unabdingbar, diese Resultate *rechtzeitig*, also innerhalb mehr oder weniger "harter" Zeitschranken, bereitzustellen. Ein richtiger Stellbefehl an ein Kühlventil, eine Sekunde zu spät geliefert, kann unter Umständen dieselben Folgen haben wie ein falscher.

Vor allem das erstere Kriterium macht den Einsatz parallelverarbeitender Computersysteme für Echtzeitsysteme äußerst attraktiv. Die Entwicklung der vergangenen Jahre zeigt diese Tendenz sehr deutlich; anstelle eines sehr großen und teuren einzelnen Prozeßrechners werden heutzutage vorwiegend mehrere, über verschiedenste Kommunikationsmedien gekoppelte Mikroprozessoren eingesetzt. Die einzelnen Prozessoren eines solchen verteilten Systems werden übrigens gerne *Nodes* genannt.

Diese Entwicklung hat natürlich auch positive Auswirkungen auf die Betriebssicherheit derartiger Systeme; bei einer entsprechend ausgelegten Software kann der (hardwaremäßige) Ausfall einer oder sogar mehrerer Komponenten ohne katastrophale Folgen bleiben (Schlagwort *Fault Tolerant Computing*).

Obwohl durch den Einsatz paralleler Prozessoren eine bedeutende Steigerung der Ausführungsgeschwindigkeit möglich ist, löst dies nicht das Problem der Rechtzeitigkeit von Verarbeitungsergebnissen. Komplexe zeitliche Abhängigkeiten von Verarbeitungsschritten untereinander bringen es mit sich, daß trotz immer größer werdender (Gesamt-)Prozessorleistung gewisse Zeitschranken nicht unterschritten

werden können. Das Problem der *Timing Constraints* bedarf daher einer anderen Lösungsmethode. Im Prinzip existieren hier zwei verschiedene "Philosophien":

- **Deterministische (Statische) Modelle**
  Dieser Ansatz basiert auf der Hypothese von (mehr oder weniger exakt) vorausbestimmbaren Ereignissen bzw. deterministischen Verarbeitungen. Darauf aufbauende Echtzeit-Software hat den unleugbaren Vorteil, die Einhaltung bestimmter Zeitbedingungen gewissermaßen a priori garantieren(!) zu können; siehe z.B. **Kop89**.

  Der Nachteil dieser Methode liegt jedoch darin, daß nicht alle technischen Prozesse tatsächlich in das deterministische Schema passen. Außerdem bringt es die in praktischen Realisierungen üblicherweise nötige synchrone Verarbeitung mit sich, daß die nötigen (Hardware-)Resourcen in Hinblick auf den Worst Case überdimensioniert (und daher teuer) und im Normalbetrieb relativ schlecht ausgenutzt sind.

- **Nichtdeterministische (Dynamische) Modelle**
  Dieser für Echtzeitsysteme sehr verbreitete Ansatz basiert auf der Ablaufsteuerung durch indeterministische Ereignisse. Ein für die Struktur einer darauf aufbauenden Echtzeit-Software äußerst unangenehme Nachteil liegt aber darin, daß es hier keine einfache Möglichkeit gibt, die Einhaltung gewisser Zeitschranken garantieren zu können (obwohl diese Tatsache in der einschlägigen Industrie schlichtweg ignoriert(!) wird). Auch die in einer derartigen Software gerne eingesetzte (und normalerweise explizit auszuprogrammierende) Überwachung der Einhaltung von Zeitschranken ist an sich keine wirkliche Alternative zu einer Garantie, wie sie statische Systeme a priori bieten können.

  Der Vorteil der nichtdeterministischen Methode liegt natürlich in der "organischen" Anpassung an nichtdeterministische technische Prozesse und in der im Regelfall besseren Auslastung der vorhandenen Hardware-Resourcen.

Welcher der beiden "Philosophien" auch der Vorzug eingeräumt wird; äußerst nützlich wäre ein Tool (eben der VTA), welches parallel zum Betrieb eines Echtzeitsystems (Timing-)Informationen betreffend frei definierbare Ereignisse (in diesem Kontext *Events* genannt) sammelt und auswertet. Im Falle deterministischer Systeme könnte z.B. durch die Beobachtung des Timings der externen Ereignisse verifiziert werden, ob die deterministische Spezifikation mit der Realität übereinstimmt. Bei nichtdeterministischen Systemen wäre es möglich, das Timing der Echtzeit-Software selbst kontinuierlich zu beobachten und zu analysieren, oder sogar die "Dimensionierung" dynamischer Datenstrukturen zu überprüfen. In jedem Fall könnte letztlich die Einhaltung gewisser Zeitschranken über längere Zeit hinweg kontrolliert werden. Eine Zusammenfassung der im Rahmen dieses Projektes primär angepeilten Einsatzmöglichkeiten des VTAs findet sich im Abschnitt *Projektziele* (Seite 2-4).

1.2 Wissenschaftliche Fragestellungen

Im Rahmen dieses Projektes sollen Fragestellungen beantwortet werden, die letztlich die Realisierung eines praktisch verwendbaren Tools ermöglichen; ein Prototyp eines *Versatile Timing Analyzers* (VTAs) ist das Endziel des gegenständlichen Projektes. Konkret sollen vor allem folgende Fragen untersucht werden:

(1) **Auf welche Weise kann parallel zum Normalbetrieb eines verteilten Echtzeitsystems eine Erfassung frei definierbarer Ereignisse (Events) erfolgen?**

Hier sind unter anderem folgende Dinge zu beachten:

(a) Vernünftiger Kompromiß zwischen vertretbarem Aufwand für den VTA einerseits und zulässiger Beeinflussung des Echtzeitsystems andererseits. Hierzu ist zu bemerken, daß übliche Verfahren des Monitorings (kleine) Veränderungen in der Echtzeit-Software vornehmen (*Instrumentierung*), die natürlich die Ausführungszeit verlängern.

(b) Die Events sollen frei definierbar, also nicht nur auf bestimmte Betriebssystem-Primitiven o.ä. beschränkt sein.

(c) Die Definition von Events soll dynamisch, also während des Betriebs des Echtzeitsystems möglich sein.

(d) Bei Eintreten eines Events sollen Timing- und andere Informationen (ebenfalls frei definierbar!) aus dem Echtzeitsystem erfaßt werden.

(e) Möglichst geringer (am besten gar kein) Zusatzaufwand bei der Software-Entwicklung für das Echtzeitsystem selbst.

(f) Zuverlässige und effiziente (zeitliche) Erfassung der Events, trotz der in einem verteilten System auftretenden Probleme mit der fehlenden globalen Zeit bzw. des unbestimmten globalen Zustandes.

(g) Möglichst universelle Verwendbarkeit des VTAs, v.a. keine Einschränkung auf eine spezielle Hard/Software-Architektur des Echtzeitsystems.

(2) **Wie kann eine Abstraktion (im Endeffekt also eine Datenreduktion) der riesigen Menge von Information erfolgen?**

Dieser Punkt ist bei allen Monitoring-Systemen von zentraler Bedeutung. Der Benutzer des VTAs sollte genau jene Information abrufen können, die er benötigt, und nicht mit großen Mengen unnötiger Daten "überfüttert" werden. Eine in diesem Zusammenhang erfolgversprechende Idee ist die Einführung verschiedener Abstraktionsebenen, etwa Statement Level, Procedure Level, Task Level, Node Level, System Level. In jeder Ebene sind nur spezifisch relevante Informationen zugänglich; durch den "Abstieg" in

darunterliegende Levels ist jedoch eine detailliertere Betrachtung ("Zooming") möglich.

Darüberhinaus ist eine möglichst frühzeitige und effiziente Datenreduktion natürlich auch in Anbetracht der endlichen Verarbeitungskapazität des VTAs angebracht.

Zu beachten ist unter anderem:

(a) Definition geeigneter Abstraktionsebenen.

(b) Flexible und trotzdem effiziente Methode zur Definition und Erfassung von komplexen (also "zusammengesetzten") Events.

(c) Flexible Methode zur Formulierung (von komplexen) Meßgrößen.

(d) Flexible Möglichkeit zur Spezifikation von Aktionen, die bei Eintreten eines Events ausgeführt werden sollen.

(e) Lösung der Probleme, die durch zusammengesetzte Ereignisse in einem verteilten System verursacht werden (Probleme betreffend die globale Zeit, Transmission Delays, ...).

(3) **In welcher Art und Weise können die gesammelten Informationen analysiert und vor allem dargestellt werden?**

Dieser Punkt ist ebenfalls von weitreichender Bedeutung. Besonders wichtig ist die Definition relevanter Meßgrößen (z.B. Interarrival Times, Durations, usw.) und die Festlegung von aussagekräftigen (etwa statistischen) Kenngrößen. Darauf aufbauend können geeignete Darstellungsmethoden gesucht werden; gerade im Zeitalter der leistungsfähigen Graphik-Workstations bieten sich hier einige Möglichkeiten an.

Besonders wichtig sind aber auch Möglichkeiten zur weitergehenden Analyse der erfaßten Daten. Ein Beispiel wäre die Realisierung einer Methode zur automatischen Erstellung einer Spezifikation für einen bestimmten technischen Prozeß (auf Basis der durch den VTA erfaßten konkreten Meßdaten). Ebenfalls interessant wären Korrelationsuntersuchungen betreffend verschiedene Ereignisse oder Verfahren zur Untersuchung der Ursachen für Grenzwertüberschreitungen gewisser Meßgrößen. Hier ergeben sich natürlich auch Anwendungsmöglichkeiten für Expertensysteme.

Unter anderem sind also folgende Dinge zu beachten:

(a) Definition von relevanten Meßgrößen, Festlegung geeigneter Kenngrößen derselben.

(b) Entwicklung geeigneter (graphischer) Darstellungsverfahren.

(c) Definition weitergehender Datenanalysen unter besonderer Berücksichtigung der Erweiterbarkeit(!) durch neue Verfahren.

1.3 Stand der Forschung

Das Studium der einschlägigen internationalen Fachliteratur zeigt überaschenderweise wenig Aktivitäten auf dem umrissenen Gebiet: Es konnten bis jetzt keine speziellen Arbeiten über die Timing-Analyse in verteilten Echtzeitsystemen gefunden werden.

Es gibt aber eine große Anzahl von Artikeln auf dem Gebiet des (Performance-)Monitorings von "gewöhnlichen" verteilten Systemen, die hauptsächlich im Kontext des *Debuggings von paralleler Software* auftauchen; ein Überblick ist etwa in **Mcd89** zu finden, theoretische Aspekte werden in **Mar90** oder **Spe88** behandelt. Die Zielsetzungen für derartige Systeme unterscheiden sich jedoch fundamental von denen des VTAs. Während bei traditionellen verteilten Systemen Maße wie mittlere Prozessorauslastung, mittlerer Grad der Parallelität usw. durchaus aussagekräftig sind, helfen sie bei Echtzeitsystemen wenig. Die Überschreitung einer Zeitbedingung kann durch Mittelwertbetrachtungen (allein) nicht erfaßt werden. Hierfür sind andere (noch festzulegende) Maße (wie z.B. die Zeit zwischen derartigen Überschreitungen) wichtig, die in traditionellen Systemen ohne Bedeutung sind. Die erwähnten Arbeiten sind daher für das vorliegende Projekt nur auf den "unteren" Ebenen (1) und teilweise auch (2) relevant.

Erwähnenswert ist vielleicht, daß Funktionen für Zeitmessungen traditionell von (guten) Logikanalysatoren und In-Circuit - Emulatoren bereitgestellt werden, allerdings nur auf dem (untersten) Statement Level. Derartige Möglichkeiten sind jedoch bei weitem nicht flexibel und vor allem erweiterbar genug, um als Basis für den VTA dienen zu können.

In Hinblick auf die im vorigen Abschnitt festgelegten wissenschaftlichen Fragestellungen ergibt sich nun etwa folgendes Bild:

(1) Ad: *Auf welche Weise kann parallel zum Normalbetrieb eines verteilten Echtzeitsystems eine Erfassung frei definierbarer Events erfolgen?*

Betreffend diesen Problemkreis ist relativ viel brauchbare Literatur nachzuweisen. Dies liegt natürlich daran, daß auch die zitierten Performance-Monitore für "gewöhnliche" verteilte Systeme parallel zum Normalbetrieb des "Untersuchungsobjektes" Information sammeln müssen. Die Möglichkeiten reichen von reiner Software-Instrumentierung (→ relativ starke Beeinflussung des "Probanden") bis zu aufwendigen, Logikanalysator-ähnlichen Hardwaremechanismen (→ geringe Beeinflussung). Einige Verfahren sind z.B. in **Ara88, Bat88, Bha87, Hab90, Rub88, Tsa90** näher erläutert.

Bezogen auf die im vorigen Abschitt umrissenen Punkte ergibt sich etwa folgender Status der Forschung:

(a) Ein Kompromiß zwischen vertretbarem Aufwand und zulässiger Beeinflussung des Echtzeitsystems wird in der Regel durch eine Kombination von Hard- und Softwaremechanismen (hybride Verfahren) erzielt. Beispiele sind in **Bha87, Hab90, Rub88, Tsa90** zu finden.

(b) Wegen der im Vergleich zum VTA eingeschränkten Erfordernisse in bezug auf die auszuwertenden Events finden sich in der Literatur überwiegend Systeme (**Hab90, Joy87, LeB90, Mil90, Tsa90**), die nur bestimmte System-Events (Task_Create, Task_Suspend, ...) oder Profiling-Events (Function Entry, Function Exit) zulassen. Frei definierbare Events werden z.B. in **Ara88, Rub88, Bha87** unterstützt.

(c) Das Runtime Setting von Events (also die Möglichkeit zur Definition neuer Events ohne Recompilation der Echtzeit-Software) ist eher eine Ausnahme (**Ara88, Rub88**).

Bei dieser Gelegenheit ist zu bemerken, daß einige der in der Literatur beschriebenen Verfahren sogar die Phase der Event-Aufzeichnung und der eigentlichen Auswertung trennen (**Ker87, LeB90**).

(d) Die meisten Systeme erfassen bei Eintreten eines Events den Zeitpunkt und eventuell einige (mehr oder weniger fixe) Zusatzinformationen (z.B. Task-IDs). Im Falle der Shared-Memory Architektur in **Ara88** werden aber auch weitergehende Möglichkeiten angeboten; dies ist natürlich bei stark auf (reiner) Instrumentierung aufbauenden Systemen relativ einfach möglich.

(e) Ein gewisser Zusatzaufwand bei der Software-Entwicklung für das Echtzeitsystem selbst wird üblicherweise in Kauf genommen. Instrumentierungen werden dabei in der Regel in das Betriebssystem oder in Librarys fix eingebaut (bei eingeschränkten Möglichkeiten der Eventdefinition, z.B. **Hab90, Joy87**), oder "händisch" (zur Übersetzungszeit) in den Code aufgenommen (**Bha87**).

(f) Die meisten Systeme gehen von einem globalen Clock aus; in **Hab90** wird eine Methode zur Synchronisation der lokalen Clocks eingesetzt. Der Thematik *Clock-Synchronisation in verteilten Systemen* wird in der Literatur sehr ausführlich Beachtung geschenkt, siehe z.B. **Arv89, Kop87, Lam78.**

(g) Die meisten der (in den bisherigen Punkten) leistungsfähigsten Systeme wie **Ara88, Bha87, Rub88** setzen spezialisierte Hard/Software-Architekturen für die zu überwachenden verteilten Systeme voraus. Relativ flexibel erscheint jedoch das **Hab90** zugrundeliegende Konzept.

Die Hauptschwierigkeit im Zusammenhang mit dem VTA liegt darin, die umrissenen Möglichkeiten in einem homogenen System zu vereinen, ohne eine zu restriktive Festlegung auf eine bestimmte Hard/Softwarestruktur des Echtzeitsystems in Kauf nehmen zu müssen. Besondere Bedeutung kommt

natürlich einer möglichst geringen Beeinflussung des Echtzeitsystems durch das parallel zum Normalbetrieb erfolgende Monitoring zu. Ebenfalls wichtig ist, daß in geographisch weit verteilten Systemen ein globaler Clock nicht günstig erscheint, d.h., daß ein geeignetes Verfahren zur Clock–Synchronisation zum Einsatz kommen sollte.

(2) Ad: *Wie kann eine Abstraktion (im Endeffekt also eine Datenreduktion) der riesigen Menge von Information erfolgen?*

Auch hier finden sich einige recht brauchbare Arbeiten aus dem Kontext des *Event–based Debuggings*. Vor allem der *Event Based Behavioral Abstraction Approach* (**Bat83, Bat88**), der auf einer Sprache zur Formulierung komplexer Events (*Event Definition Language, EDL)* basiert, scheint für das vorliegende Projekt recht brauchbar zu sein.

Im einzelnen ergibt sich etwa folgendes Bild:

(a) Die Notwendigkeit der Einführung geeigneter Abstraktionsebenen wird in den meisten Arbeiten formuliert und auch berücksichtigt. Meist wird hierbei, ausgehend von der höchsten Ebene des Gesamtsystems über die Ebene der Prozessoren, Tasks und schließlich Funktionen die unterste Ebene der Statements erreicht (**Hab90, LeB90, Mil90**). Eine andere, sehr flexible Methode zur Abstraktion wird in **Bat83, Bat88** und auf etwas andere Art und Weise in **Joy87** verwendet. Hier können beliebige Informationen (konkret Events) zu "Einheiten" zusammengefaßt werden, deren "interner" Aufbau dann nicht mehr relevant ist (– *Information Hiding*).

(b) Die Definition und Erfassung von komplexen (also "zusammengesetzten") Events wird, wie schon erwähnt, in **Bat83, Bat88** (mit Hilfe von EDL) und in **Rub88** unterstützt.

(c) Eine flexible Methode zur Formulierung (komplexer) Meßgrößen ist durch die Festlegung geeignert Aktionen, die bei Auftreten eines Events ausgeführt werden, gegeben; siehe dazu Punkt (d).

(d) Die Arbeiten **Bat83, Bat88, Bha87** gehen von einem Event – Action Modell aus. Hierbei kann sowohl ein Event als auch eine Reihe von Aktionen, die beim Eintreten eines bestimmten Events (im Monitoring–System!) ausgeführt werden sollen, spezifiziert werden.. Auf diese Art und Weise sollte auch das Problem der Formulierung von Meßgrößen zu lösen sein.

(e) Die meisten Artikel setzen einen globalen Clock voraus, wodurch die totale Ordnung von Events kein großes Problem darstellt. In einem System mit lokal synchronisierten Clocks bzw. in lose gekoppelten verteilten Systemen ist jedoch das Problem des Erkennens des Eintretens eines zusammengesetzten Events bei weitem nicht trivial, siehe z.B.

Lam78. Einige diesbezügliche Aspekte werden in **Bat83, Bat88** diskutiert.

Im Rahmen des vorliegenden Projektes soll ein den spezifischen Problemen bei Echtzeitsystemen Rechnung tragendes Abstraktionsmodell (bzw. ein entsprechender Abstraktionsmechanismus!) vermutlich auf der Basis einer adaptierten Event Definition Language gefunden werden. Besonderes Augenmerk muß dabei auf die Bereitstellung von "Sprachmitteln" gelegt werden, die den "Transport" von relevanter Information (v.a. Meßgrößen, z.B. die Zeit zwischen dem Eintreten zweier Events) in höhere Ebenen gestatten. Diese Sprache muß so flexibel ausgelegt werden, daß eine weitgehende Entkopplung dieses Problemkreises (2) vom folgenden Problemkreis (3) stattfindet.

Von ganz zentraler Bedeutung ist natürlich der Aspekt der Realisierung eines verteilten Event Recognizers, die, wie schon erwähnt, einer sehr gründlichen Untersuchung bedarf. Die Hauptschwierigkeiten dabei treten im Zusammenhang mit zusammengesetzten Events, deren konstituierende (Teil-)Events von verschiedenen Prozessoren stammen, auf.

(3) Ad: *In welcher Art und Weise können die gesammelten Informationen analysiert und vor allem dargestellt werden?*

Betreffend diesen Problemkreis läßt sich nur sehr wenig verwendbare Literatur nachweisen; einige Ansätze in Richtung Real Time Systems sind in **Bha87** zu finden. Im Kontext der "gewöhnlichen" verteilten Systeme gibt es natürlich einige interessante Ideen; diese sind für den VTA allerdings wenig bis gar nicht brauchbar.

Bezüglich der Darstellung der gesammelten Information gibt es z.B. zwei prinzipiell unterschiedliche Ideen, die grob mit *Zeitdarstellungen* und *Animation* umrissen werden können. Bei Zeitdarstellungen (**Bha87, Fow88, Hab90, Ker87, LeB90, Mil90**) handelt es sich um Darstellungen gewisser Kenngrößen in Koordinatensystemen, bei denen eine Koordinate die Zeit repräsentiert; die einfachste Form sind natürlich die gewöhnlichen Timing-Diagramme. Bei Animationen (**Joy87, Rub88, Hab90, Hou88, Soc88**) wird hingegen mit verschiedensten zeitlich veränderlichen Darstellungen operiert; eine solche Darstellung ist daher immer nur eine Momentaufnahme des aktuellen Systemzustandes.

Es zeigt sich also etwa folgender Status der Forschung:

- Erste Ansätze betreffend die Festlegung von Kenngrößen für relevante Meßgrößen finden sich in **Bha87**.

- Ideen für graphische Darstellungsverfahren können u.U. von den Arbeiten über Zeitdarstellungen (s. oben) stimuliert werden (**Bha87, Fow88, Hab90, Ker87, LeB90, Mil90**). Auf Animation basierende Techniken dürften für Echtzeitsysteme ausscheiden.

- Was die weitergehenden Datenanalysen betrifft, konnten bis jetzt keine praktisch brauchbaren Vorarbeiten gefunden werden. Einige konzeptuelle Ansätze im Zusammenhang mit der automatischen Generierung von Verhaltensmodellen für parallele Software auf der Basis der Event-History werden in Mil90 und in Bat88 vorgestellt.

Der hier umrissene Problemkreis stellt ohne Zweifel jenes Gebiet dar, in dem praktisch ausschließlich völliges Neuland beschritten werden muß. Als wichtigste Aufgabe ist hierbei die Festlegung von relevanten Meßgrößen anzusehen; einfache Beispiele dafür wären Interarrival Times oder Durations. Ebenso wichtig ist die Festlegung aussagekräftiger Kenngrößen dafür; ("gewöhnliche") Mittelwerte sind, wie schon erwähnt, nicht sehr brauchbar.

Als zweiter Schritt sind Darstellungsverfahren zu suchen, die dem Benutzer möglichst nur jene Information liefern, die er benötigt.

Ebenfalls eingehender Untersuchung bedürfen weitergehende Datenanalysen. So muß etwa festgestellt werden, inwieweit die in der Literatur zu findenden Ansätze für den VTA brauchbar sind bzw. wie schließlich brauchbare Verfahren (eventuell auf Basis eines Expertensystems) aussehen können.

Der Antragsteller kann auf einige (auch industrielle) Erfahrungen im Zusammenhang mit spezieller Hardware, hardwarenaher Echtzeitprogrammierung und nicht zuletzt Clock-Synchronisation in verteilten Systemen bzw. verwandten Themen verweisen. Ein ganz wesentlicher Forschungsschwerpunkt liegt zur Zeit auf der theoretischen (mathematischen) Untersuchung der Performance nichtdeterministischer Echtzeitsysteme, insbesondere der Definition von in diesem Zusammenhang relevanten Kenngrößen.

Vor allem die zuletzt angedeuteten Forschungen sind als theoretische Vorarbeiten zu diesem Projekt anzusehen. Einige der dabei gewonnenen Resultate verlangen geradezu nach einem "Meßwerkzeug" wie dem VTA, mit dem eine praktische Verifikation derselben möglich ist; sie sind im Endeffekt der eigentliche Stimulus für das gegenständliche Projekt.

1.4 Literatur

Ara88 Aral, Z., Gertner, I., *"High Level Debugging in Parasight"*, Proc. ACM SIGPLAN and SIGOPS Workshop on Parallel and Distributed Debugging (1988), 151-162

Arv89 Arvind, K., *"A New Probabilistic Algorithm for Clock Synchronization"*, Proc. Real-Time Systems Symposium IEEE (1989), 330-339

Bat83 Bates, P., Wileden, J.C., *"High-Level Debugging of Distributed Systems: The Behavioral Abstraction Approach"*, Journal of Systems and Software 3, (1983), 255-264

Bat88 Bates, P., *"Debugging Heterogeneous Distributed Systems Using Event-Based Models of Behavior"*, Proc. ACM SIGPLAN and SIGOPS Workshop on Parallel and Distributed Debugging (1988), 11-22

Bha87 Bhatt, D., Ghonami, A., Ramanujan, R., *"An Instrumented Testbed for Real-Time Distributed Systems Development"*, Proc. Real-Time Systems Symposium IEEE (1987), 241-250

Fow88 Fowler, R.J., LeBlanc, T.J., Mellor-Crummey, J.M., *"An Integrated Approach to Parallel Program Debugging and Performance Analysis on Large-Scale Multiprocessors"*, Proc. ACM SIGPLAN and SIGOPS Workshop on Parallel and Distributed Debugging (1988), 163-173

Hab90 Haban, D., Wybranietz, D., *"A Hybrid Monitor for Behavior and Performance Analysis of Distributed Systems"*, IEEE Trans. Soft. Eng., Vol. 16, No. 2 (Feb. 1990), 197-211

Hou88 Hough, A., Cuny, J.E., *"Initial Experiences with a Pattern-Oriented Parallel Debugger"*, Proc. ACM SIGPLAN and SIGOPS Workshop on Parallel and Distributed Debugging (1988), 195-205

Joy87 Joyce, J., Lomow, G., Slind, K., Unger, B., *"Monitoring Distributed Systems"*, ACM Trans. Comput. Syst., Vol. 5, No. 2, (May 1987), 121-150

Lam78 Lamport, L., *"Time, Clocks and the Ordering of Events in a Distributed System"*, Comm. ACM, Vol. 21, No. 7, (July 1978), 558-565

LeB90 LeBlanc, T.J., Mellor-Crummey, J.M., Fowler, R.J., *"Analyzing Parallel Program Executions Using Multiple Views"*, Journal of Parallel and Distributed Computing 9, (1990), 203-217

Ker87 Kerola, T., Schwetman, H., *"Monit: A Performance Monitoring Tool for Parallel and Pseudo-Parallel Programs"*, Proc. ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems (1987), 163-174

Kop87 Kopetz, H., Ochsenreiter, W., *"Clock Synchronization in Distributed Real Time Systems"*, IEEE Trans. Comput., Vol 36, No. 8, (August 1987), 933-940

Kop89 Kopetz, H., Damm, A., Koza, C., Mulazzani, M., Schwabl, W., Senft, C., Zainlinger, R., *"Distributed Fault-Tolerant Real-Time Systems: The Mars Approach"*, IEEE MICRO, (Feb. 1989), 25-40

Mar90 Marinescu, D.C., Lumpp, J.E.Jr., Casavant, T.L., *"Models for Monitoring and Debugging Tools for Parallel and Distributed Software"*, Journal of Parallel and Distributed Computing 9, (1990), 171-184

Mcd89 McDowell, C.E., Helmbold, D.P., *"Debugging Concurrent Programs"*, ACM Comput. Surv., Vol. 21, No. 4, (December 1989), 593-622

Mil90 Miller, B.P., Clark, M., Hollingsworth, J., Kierstead, S., Lim, S., Torzewski, T., *"IPS-2: The Second Generation of a Parallel Program Measurement System"*, IEEE Trans. Par. and Distrib. Syst., Vol. 1., No. 2, (April 1990), 206-217

Rub88 Rubin, R.V., Rudolph, L., Zernik, D., *"Debugging Parallel Programs in Parallel"*, Proc. ACM SIGPLAN and SIGOPS Workshop on Parallel and Distributed Debugging (1988), 216-225

Sch83 Schrott, G., Tempelmeier, T., *"Monitoring of Real Time Systems by a Seperate Processor"*, Proc. 12th IFAC/IFIP Workshop on Real Time Programming (1983), 69-79

Seg85 Segall, Z., Rudolph, L., *"PIE: A Programming and Instrumentation Environment for Parallel Processing"*, IEEE Software, (Nov. 1985), 22-37

Soc88 Socha, D., Bailey, M.L., Notkin, D., *"Voyeur: Graphical Views of Parallel Programs"*, Proc. ACM SIGPLAN and SIGOPS Workshop on Parallel and Distributed Debugging (1988), 206-215

Spe88 Spezialetti, M., Kearns, J.P., *"A General Approach to Recognizing Event Occurrences in Distributed Computations"*, Proc. 8th Int. Conf. on Distrib. Comp. Syst., (1988), 300-307

Sto88 Stone, J.M., *"A graphical representation of concurrent processes"*, Proc. ACM SIGPLAN and SIGOPS Workshop on Parallel and Distributed Debugging (1988), 226-235

Tsa90 Tsai, J.J.P, Fang, K., Chen, H., Bi, Y., *"A Noninterference Monitoring and Replay Mechanism for Real-Time Software Testing and Debugging"*, IEEE Trans. Soft. Eng., Vol. 16, No. 8, (Aug. 1990), 897-916

Wyb88 Wybranietz, D., Haban, D., *"Monitoring and Performance Measuring Distributed Systems During Operation"*, Proc. ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems (1988), 197-206

2. Projektziele/Projektdurchführung

Aufgabe dieses Abschnittes ist es, die konkreten Ziele des vorliegenden Projektes sowie dessen praktische Durchführung zu beschreiben. Es sollte hierbei klar sein, daß die Beantwortung der vorher umrissenen wissenschaftlichen Fragestellungen eine dafür notwendige Voraussetzung darstellt.

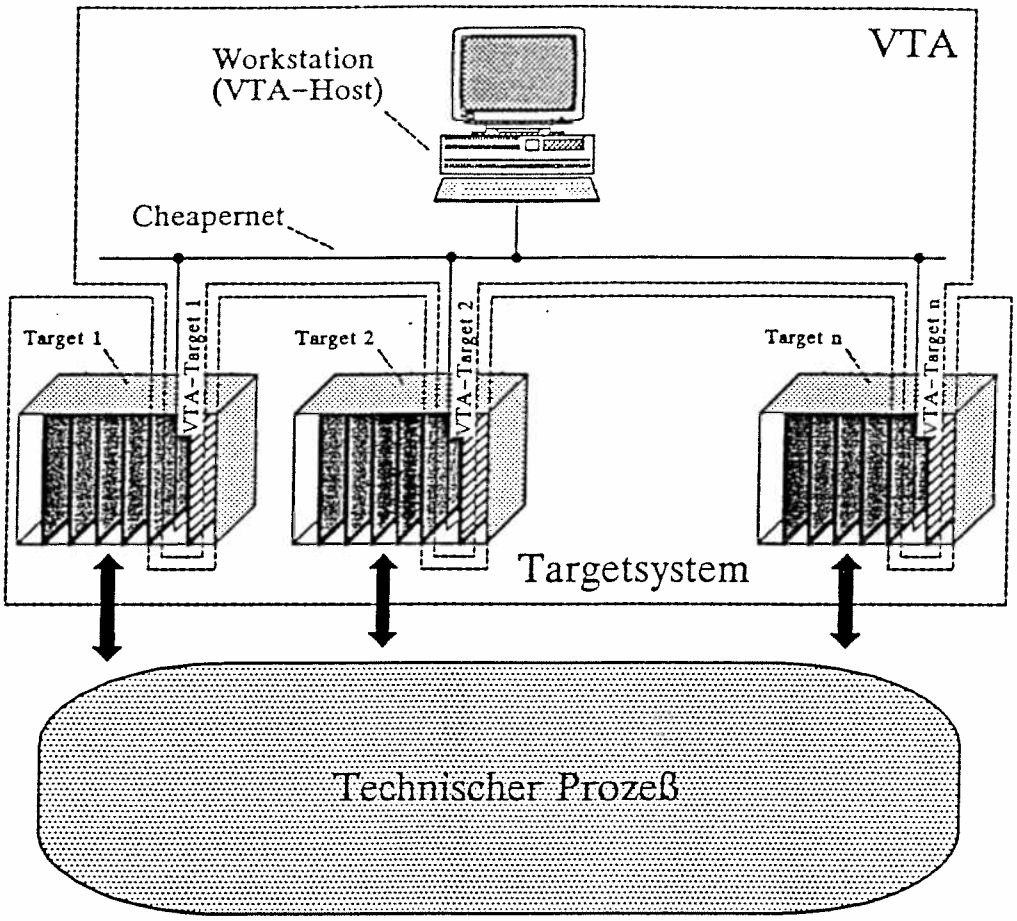
2.1 Projektziele

Der *VTA* (*Versatile Timing Analyzer*) ist ein System zur Überwachung bzw. statistischen Auswertung des zeitlichen Auftretens frei definierbarer *Events* in einem beliebigen verteilten VME-Multiprozessor Echtzeitsystem, das im folgenden als *Targetsystem* bezeichnet wird.

2.1.1 Konfiguration

Der VTA ist für Targetsysteme gedacht, die aus einem oder mehreren kompletten VME-Systemen (Racks mit einer Backplane und einem oder mehreren CPU-Boards) bestehen. Wie bzw. ob die einzelnen Targets untereinander verbunden sind (Profibus, MAP, ...), ist unerheblich.

Das VTA-System selbst besteht aus zwei Komponenten, einer Workstation (*VTA-Host*) als Bedienungselement und einem oder mehreren speziellen VME CPU-Boards (*VTA-Targets*). Letztere werden in einen freien Slot eines jeden (relevanten) Targets gesteckt, gehören aber von ihrer Aufgabe her nicht zum eigentlichen Targetsystem. Die Kopplung der einzelnen VTA-Komponenten untereinander erfolgt über eine Cheapernet-Verbindung (IEEE 802.3). Das folgende Bild zeigt den prinzipiellen Aufbau:



Prinzipieller Aufbau des VTAs

2.1.2 Funktionalität

Ein *Event* wird im wesentlichen durch die Angabe einer bestimmten Instruktion in der Targetsystem-Software, eventuell ergänzt durch zusätzliche Nebenbedingungen, definiert. Ein solches Event tritt nun genau dann ein, wenn ein Prozessor das spezifizierte Statement ausführen will und die Nebenbedingungen zu diesem Zeitpunkt erfüllt sind. Neben diesen Statement Events können aber auch eine ganze Reihe spezieller Events, wie z.B. das Dispatching eines bestimmten Tasks oder das Erreichen gewisser Zeitpunkte aufgesetzt werden. Die Nebenbedingungen können etwa einen bestimmten Prozessor, einen bestimmten Task oder gewisse Werte von Variablen fordern.

Eine qualitativ gänzlich andere Art von Nebenbedingungen ermöglicht es, das Eintreten eines Events vom vorhergehenden Eintreten anderer Events abhängig zu machen; hierbei handelt es sich um einen Abstraktionsmechanismus auf der Basis einer Event Definition Language, die die Formulierung von komplexen (zusammengesetzten) Events erlaubt.

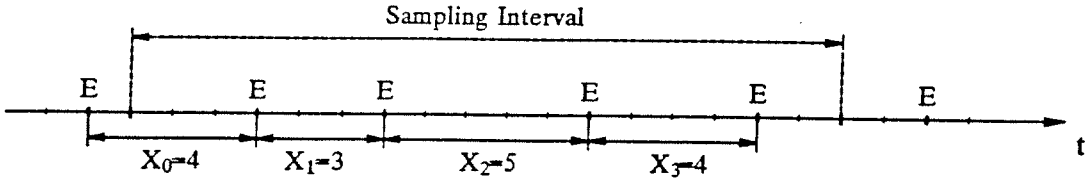
Solche Events werden nun zur Festlegung von *Quantities* (*Meßgrößen*) herangezogen. Einfache Beispiele dafür sind etwa *Interarrival Times*, also Zeitintervalle zwischen dem Eintreten eines (zyklischen) Events E, oder *Durations*, d.h., Zeitabschnitte zwischen dem Eintreten der Events A und B. Auch die Häufigkeit des Eintretens eines Events E innerhalb eines gewissen Zeitabschnittes (*Eventcounts*)

oder der Wert einer bestimmten Variablen beim Eintreten eines Events E gehören dazu. Aus solchen einfachen Quantities können dann auch komplexere Dinge (wie z. B. Summen von Durations) und sogar zusätzliche Events (etwa das Überschreiten eines Grenzwertes durch eine Quantity) gewonnen werden.

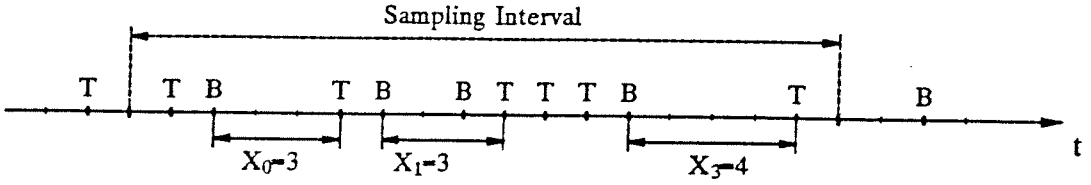
Jeder Quantity kann ein zyklisches (u.U. recht komplexes) *Sampling Interval* (*Auswertungsintervall*) zugeordnet werden. Der VTA überwacht den Ablauf der Software am Targetsystem und hat daher im Normalfall am Ende eines Sampling Intervals eine ganze Folge von beobachteten Werten für eine Quantity zur Verfügung. Diese wird dazu herangezogen, um eine Reihe von (statistischen) Parametern wie Mittelwert, Varianz, Maximalwert, Minimalwert, Verteilungsfunktion (Histogramm), usw. zu berechnen.

Die folgenden Diagramme zeigen einige Beispiele für mögliche Meßgrößen:

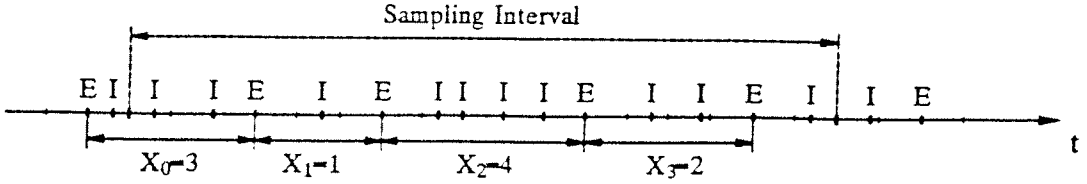
- *Interarrival Times*
  Eine Interarrival Time wird durch ein einzelnes Event E bestimmt:



- *Durations*
  Dies ist die (nicht-negative) Differenz zwischen dem Auftreten von T und B, wobei B vor T kommen muß:



- *Eventcounts*
  Im Gegensatz zu den bereits vorgestellten Interarrival Times handelt es sich hier um die Erfassung der Anzahl von Events I zwischen je zwei Events E:



2.1.3 Bedienung und Funktionsweise

Das Aufsetzen von Events und die Festlegung von Quantities und Sampling Intervals erfolgt (hauptsächlich) über die Workstation, und zwar zur Laufzeit des Targetsystems. Der VTA erlaubt das Setzen von Statement Events auf Hochsprachenebene, rechnet also einerseits mit dem Vorhandensein der entsprechenden (Source-)Files auf der Workstation und andererseits mit einer "modifizierbaren" (im RAM befindlichen) Software im Targetsystem. Abhängig von der Hardware der einzelnen Targets ist jedoch nur ein minimaler Zusatzaufwand bei der Erstellung der Targetsoftware erforderlich (maximal das Linken eines *VTA Support Packages*). Es ist aber selbstverständlich auch möglich, gewisse (v. a. extrem zeitkritische) Events schon bei der Entwicklung der Targetsystem-Software einzubinden ("klassische" Instrumentierung).

Die Workstation (VTA-Host) hat primär die Aufgabe, das Human Interface des VTAs bereitzustellen und die CPU-intensiven Berechnungen duchzuführen. Die Funktionen des Human Interfaces betreffen die allgemeine Kontrolle des VTAs, das Aufsetzen von Events, die Festlegung von Quantities und Sampling Intervals sowie die (graphische) Ausgabe der Meßergebnisse (Tabellen, Histogramme, Kurven, ...). Durch ein benutzerfreundliches Design wird eine maximale Funktionalität bei gleichzeitig minimalem Bedienungsaufwand erreicht.

Die eigentliche Überwachung der einzelnen Targets ist Aufgabe der VTA-Targets. Ihnen obliegt etwa das eigentliche Setzen der Events im RAM des jeweiligen Targets und das (verteilte!) Monitoring der eintretenden Events. Die in diesem Zuge erfaßten Daten werden vorverarbeitet und schließlich über das Ethernet an die Workstation geschickt.

2.1.4 Einsatzmöglichkeiten

Mit dem VTA steht ein Werkzeug zur Verfügung, mit dessen Hilfe dem bei Echtzeitsystemen so kritischen Problemkreis der *Time Constraints* praktisch zu Leibe gerückt werden kann. Das zugrundeliegende Konzept stellt dabei einen vernünftigen Kompromiß zwischen vertretbarem Aufwand einerseits und zulässiger Beeinflussung der ablaufenden Targetsoftware andererseits dar.

Der VTA erlaubt es zum Beispiel, die Dauer einer zeitkritischen Befehlssequenz der Targetsoftware kontinuierlich zu überwachen und auf diese Weise Performance-Engpässe (oder Überdimensionierungen) aufzudecken. Die Besonderheit liegt dabei in der Möglichkeit, komplexe Bedingungen spezifizieren und auf diese Weise den "Scope" einer solchen Untersuchung z. B. auf einen oder mehrere bestimmte Tasks einschränken zu können. Spezielle komplexe Quantities gestatten es sogar, Entscheidungen betreffend die Verteilung von Tasks auf verschiedene Prozessoren zu verifizieren. Selbst die Dimensionierung von Speicherplatz für dynamische Datenstrukturen (Buffer, Heaps, Stacks, ...) kann überprüft werden.

Abseits von diesen für die Praxis so wichtigen Einsatzgebieten stellt der VTA ein ganz wesentliches Hilfsmittel dar, um dem Problem *"Performance-Requirements für nichtdeterministische Echtzeitsysteme"* praktisch nähertreten zu können.

Dabei geht es, grob gesagt, um die Bereitstellung von (theoretischen) Modellen, die letztlich (in einer frühen Phase der Entwicklung von Echtzeit-Software) Aussagen über die nötige Performance erlauben. "Klassische" Performance-Maße wie die mittlere CPU-Belastung usw. sind ja in Hinblick auf Time Constraints relativ bedeutungslos.

Es ist in diesem Zusammenhang zunächst sehr wichtig, Informationen über die Stimuli zu gewinnen, mit denen reale technische Prozesse ein Prozeßautomatisierungssystem konfrontieren. Mit Hilfe des VTAs ist es möglich, derartige Kenngrößen praktisch zu bestimmen bzw. zu verifizieren (es ist dazu lediglich notwendig, Events in den für die Bedienung der Prozeßperipherie zuständigen Programmteilen aufzusetzen). Die Voraussetzung ist natürlich ein an dem jeweiligen technischen Prozeß angekoppeltes Targetsystem.

Darüberhinaus können aber auch die "Auswirkungen" derartiger Stimuli, also letztlich die in der Targetsoftware ausgelösten weiteren (also weitergehenden) Aktivitäten, verfolgt werden. Auf eine derartige praktische "Beobachtung" realer Systeme kann in Hinblick auf eine vernünftige Modellbildung wohl kaum verzichtet werden. Allerdings erfordern derartige Anwendungen des VTAs mit Sicherheit erweiterte Meß- und Auswertungsmöglichkeiten; entsprechende Erfordernisse können allerdings zu diesem Zeitpunkt noch nicht festgelegt werden.

2.1.5 Erweiterungsmöglichkeiten

Durch das offene Design des VTAs können mit Hilfe von *Language Support Packages* verschiedene Programmiersprachen (zum Beispiel *C* oder *Ada*) für die Implementierung der Targetsystem-Software zugelassen werden. Ebenso erlauben verschiedene *Operating System* – und *Board Support Packages* die Unterstützung unterschiedlicher Target-Betriebssysteme (pSOS, VRTX, OS9, VxWorks, ...) bzw. verschiedener Target CPU-Boards und Prozessoren.

Mit Hilfe einer ganz speziellen Art von Board Support Packages ist es sogar möglich, die VTA-Target Software als Teil der Targetsystem-Software mitlaufen zu lassen. Anders ausgedrückt läuft die Targetsystem-Software (teilweise) auf der VTA-Target Hardware, womit sich eine sehr kostengünstige Variante für kleine Systeme (z. B. einen Prozeßmonitor) ergibt.

Im Design von VTA vorgesehen ist auch die Ausbaumöglichkeit in Hinblick auf zusätzliche Meß- und Auswertungsverfahren. So ist es in weiterer Folge geplant, die Funktionalität des VTAs durch komplexe statistische Auswertungen bis hin zu Hypothesentests zu erweitern.

Natürlich ist es im Prinzip auch möglich, andere Bussysteme (wie etwa den *Multibus*) zu unterstützen. Eine derartige Erweiterung erfordert allerdings die (Neu-)Entwicklung der jeweiligen VTA-Targets. Durch die Entwicklung einer (Logikanalysator-ähnlichen) Spezial-Hardware könnten darüberhinaus auch zusätzliche spezielle Events betreffend gewisse (Bus-)Signale (z. B. Interrupt-Leitungen!) hinzugefügt bzw. ein weniger "belastendes" Monitoring realisiert werden.

2.2 Projektdurchführung

Bevor auf die Aspekte der praktischen Durchführung des Projektes eingegangen werden kann, sollten zunächst die Gründe, die zu der vorher beschriebenen Struktur des VTAs geführt haben, vorgestellt werden.

2.2.1 Begründung der Konfiguration

Die grundsätzliche Entscheidung, primär VME-basierende Echtzeitsysteme zu unterstützen, wird durch mehrere Argumente gerechtfertigt:

- Der Marktanteil von Echtzeitsystemen, die auf dem VME-Bus basieren, ist sehr groß. Hierbei ist zu beachten, daß dem VME-Bus eine offengelegte Spezifikation (ursprünglich von Motorola, Philips, Thomson-CSF und Signetics gemeinsam initiiert) zugrundeliegt, wodurch bereits sehr viele Anbieter fertige VME-Komponenten (CPUs, Speicher, Peripherie, ...) anbieten. Es gibt auch eine eigene Fachzeitung (*VME-bus*, erscheint im Franzis-Verlag), die sich ausschließlich mit derartigen Systemen beschäftigt.

- Für die VTA-Targets können fertige VME-Komponenten verwendet werden, wodurch die Notwendigkeit einer Hardware-Entwicklung entfällt. Es ist möglich, das FMB Feature (*Force Message Broadcast*, ein sehr schnelles und universell einsetzbares Bus-Broadcasting) der von der Firma Force angebotenen CPU-Komponenten für eine effiziente Erfassung der aus dem Target kommenden Events einzusetzen.

- Da wir aus den erwähnten Gründen VME-basierende Hardware auch in den Übungen zur Vorlesung "Prozeßautomatisierung" (Pflichtveranstaltung für Informatiker im 4. Semester, TU-Wien) einsetzen, ist bei allen Mitarbeitern das entsprechende Know-How vorhanden.

- Für die erwähnten Übungen zur "Prozeßautomatisierung" stehen zwei komplette VME-Echtzeitsysteme zur Verfügung, die an verschiedenste technische Prozesse angekoppelt sind (u.a. eine Modelleisenbahn, diverse Motoren und ein Robotermodell). Diese Systeme können als (Basis-)Testumgebung für die praktische Realisierung des VTAs verwendet werden, wodurch die Anschaffung einer solchen aus Projektmitteln unterbleiben kann.

Die Software für die VTA-Targets selbst soll unter dem Betriebssystem pSOS+m (68030 Multiprozessor-Version) von SCG laufen. Für den VTA-Host kommt hingegen als Betriebssystem nur UNIX in Frage, die Benutzeroberfäche soll auf X-Windows/MOTIF basieren. Auch für diese Entscheidungen sprechen einige Punkte:

- pSOS+ ist ein sehr effizienter, ORKID-konformer Betriebssystem-Kernel, der auch unter dem Namen *VMEexec* auch von Motorola vertrieben wird. Außerdem gibt es eine relativ gute Cross-Entwicklungsumgebung unter

UNIX (auf Sun–Workstations) und (mit eingeschränkten Möglichkeiten) auf PCs (AT–286), die ein symbolisches Debugging einer entwickelten Software im Zielsystem(!) erlaubt. Die Kopplung von Entwicklungs– und Zielsystem erfolgt dabei über Ethernet oder (in der PC–Version) über eine V24–Verbindung.

- Bei den Übungen zur Prozeßautomatisierung wird das Betriebssystem pSOS+ (68000 Single–Prozessor) eingesetzt, welches funktional bis auf den Multiprozessor–Support mit pSOS+m übereinstimmt. Damit ist das entsprechende Know–How bei den Projektmitarbeitern bereits vorhanden. Auch die Anschaffungskosten für die Betriebssystemkomponenten sind kleiner: Es ist lediglich notwendig, die existierenden Lizenzen mit einem (kostengünstigen) Upgrade auf die benötigte Version zu bringen.

- Bei den Übungen wird (natürlich) die kostengünstigere PC–Version der Entwicklungsumgebung eingesetzt, da etwa die (Speicher–)Limitierungen und das langsame Downloading über die serielle Schnittstelle bei den kurzen Übungsprogrammen keine wesentlichen Probleme darstellen. Für die Entwicklung des VTAs ist diese PC–Version allerdings nicht verwendbar, wodurch die Anschaffung der Entwicklungsumgebung für eine Sun–Workstation unabdingbar ist. Da für den VTA–Host aber ohnedies eine Workstation erforderlich ist, auf der auch die Source–Files der Echtzeit–Software verfügbar sein müssen (siehe Abschnitt *Projektziele*, Seite 2–4), ist es naheliegend, diese Sun auch dafür einzusetzen.

- Die Forschungsstätte verfügt über ca. 20 leistungsfähige HP/Apollo–Workstations, für die die angesprochene Entwicklungsumgebung aber leider nicht erhältlich ist (und dies lt. Auskunft des Distributors auch nie sein wird). Nichts desto trotz können gewisse Teile des Projektes, v.a. die Dokumentationserstellung oder das Editieren auf einigen dafür bereitgestellten Apollo–Workstations durchgeführt werden. Sehr wesentlich ist auch, daß X–Windows/MOTIF auf den Apollos verfügbar ist, sodaß die primäre Entwicklung der Software für den VTA–Host ebenfalls auf den Apollos erledigt werden kann.

  Die für die Integration einer Sun in das Apollo–Netzwerk nötige NFS–Software ist bereits vorhanden. Notwendig ist allerdings die Anschaffung eines Dokument Management Systems, welches sowohl auf Sun als auch auf Apollos verwendbar ist (konret INTERLEAF); das normalerweise bei uns eingesetzte CONTEXT (Mentor Graphics) ist auf einer Sun nämlich nicht verfügbar. Auf die wesentliche Bedeutung einer vollständigen und konsistenten Dokumentation für ein derartiges Projekt braucht wohl nicht gesondert eingegangen zu werden.

2.2.2 Methodik

Bei der konkreten Durchführung des Projektes lassen sich, neben der koordinierenden Tätigkeit des Projektleiters, vier voneinander relativ unabhängige Aufgabengebiete orten:

(1) **Bereitstellung eines Ethernet-basierenden verteilten pSOS+m Multiprozessorsystems mit Clock-Synchronisation (1 Diplomand, 1 Jahr)**
Hierbei handelt es sich um die Bereitstellung der "Infrastruktur" für die VTA-Targets. Konkret müssen die Betriebssystemkomponenten pSOS+m (Kernel), pROBE+ (System State Debugger), pREPC+ (C-Interface) und pNA+ (Ethernet TCP/IP) im EPROM der VTA-Targets (Force CPU-30) installiert werden. Ebenso muß die gesamte Entwicklungsumgebung auf der Sun-Workstation installiert und das Zusammenspiel der Komponenten überprüft werden.

Die zentrale Problematik ist allerdings die Implementierung einer geeigneten Methode zur Clock-Synchronisation. Als Endergebnis dieser Bemühungen sollte ein "schlüsselfertiges" pSOS+m Multiprozessorsystem (aus Kostengründen mit nur 2 Prozessoren) bereitstehen, auf dem die Implementierung der Software für die VTA-Targets stattfinden kann.

(2) **Realisierung der VTA-Target Software (1 Univ. Ass., 1 Diplomand, je 2 Jahre)**
Dieser Punkt setzt die Beantwortung der im Abschnitt *Wissenschaftliche Fragestellungen* (Seite 1-3) in den Punkten (1) und vor allem (2) umrissenen Fragen voraus und stellt wohl eine der aufwendigsten Aufgaben dar.

(3) **Realisierung der Low-Level VTA-Host Software (1 Diplomand, 2 Jahre)**
Hier geht es um die Realisierung der nicht primär die Benutzerschnittstelle betreffenden Funktionalität des VTA-Hosts, v.a. um die Abwicklung des Protokolles mit den VTA-Targets, die Bereitstellung eines Übersetzers für die Event Definition Language und die Realisierung der erwähnten Abstraktionsmechanismen. Es handelt sich hierbei also hauptsächlich um eine sehr anspruchsvolle und zeitaufwendige Design- und Implementierungstätigkeit.

(4) **Realisierung der High-Level VTA-Host Software (Projektleiter, 1 Diplomand, 2 bzw. 1 Jahre)**
Hier soll der benutzerorientierte Teil des VTA-Hosts behandelt werden. Es ist aber nicht primär die Bereitstellung der Benutzeroberfläche selbst, sondern vor allem die Beantwortung der wissenschaftlichen Fragestellungen (3), die diesen Themenkreis extrem schwierig und daher auch zeit- und personalaufwendig machen.

Die Voraussetzung für eine weitgehende Unabhängigkeit der angesprochenen Teilgebiete ist natürlich die Entwicklung einer exakten globalen Spezifikation betreffend die Funktionalität des VTAs und, nicht zuletzt, eine enge Zusammenarbeit der Mitarbeiter gerade in den frühen Phasen der Projektdurchführung. Es bietet

sich daher folgende Projektorganisation an:

(1) Entwicklung einer funktionalen Basis-Spezifikation für den VTA durch den Projektleiter. Währenddessen können sich die weiteren Mitarbeiter in die vorhandene Literatur einlesen bzw. eine umfassende Recherche betreffend weitere Veröffentlichungen anstellen. *Dauer: 1-2 Monate.*

Parallel dazu kann ein Mitarbeiter mit der Bereitstellung des pSOS+m Multiprozessorsystems mit Clock-Synchronisation beginnen, sodaß bei der späteren Implementierung der VTA-Targets keine unnötigen Wartezeiten in Kauf genommen werden müssen. *Dauer: ca. 1 Jahr.*

(2) Abwechselnde gemeinsame Diskussion der (Basis-)Spezifikation und Bearbeitung der wissenschaftlichen Fragestellungen durch die einzelnen Projektmitarbeiter. Das Ziel dieser (sicherlich mehrstufigen) Phase ist eine exakte VTA-Spezifikation, die dann im Rahmen einer Präsentation, bei der Mitarbeiter anderer Forschungseinrichtungen eingeladen werden sollen, vorgestellt wird. Auf diese Weise können wertvolle Anregungen in einer frühen Phase des Projekts berücksichtigt werden. Für die Erstellung der notwendigen Dokumente stehen neben der aus Projektmitteln anzuschaffenden Sun auch die Apollo-Workstations der Forschungsstätte zur Verfügung. *Dauer: 6-8 Monate.*

(3) Erstellung der Dokumentation, die die Schnittstellen zwischen den vorher umrissenen Teilgebieten exakt definiert. Diese Arbeit kann größtenteils parallel erfolgen und sollte primär vom Projektleiter koordiniert werden. Auch hier stehen neben der aus Projektmitteln anzuschaffenden Sun die Apollo-Workstations der Forschungsstätte zur Verfügung. *Dauer: ca. 1 Monat.*

(4) Auf dieser Basis kann mit der Implementierungsphase begonnen werden. Zu diesem Zeitpunkt sollte auch die Infrastruktur für die VTA-Targets weit genug entwickelt sein.

Die Realisierung der VTA-Target Software erfolgt dabei primär auf der aus Projektmitteln anzuschaffenden Sun-Workstation (natürlich für das angekoppelte pSOS+m Multiprozessorsystem).

Für die VTA-Host Software stehen die Apollo-Workstations der Forschungsstätte zur Verfügung; der Übergang auf den eigentlichen VTA-Host (die Sun) ist erst ganz zum Schluß erforderlich.

Dauer: 6-8 Monate.

(5) Für die ersten Testphasen stehen, wie schon erwähnt, die beiden für die Übungen zur "Prozeßautomatisierung" eingesetzten Systeme als Testumgebung (= Echtzeitsystem) bereit. Für die Endphase des Tests ist jedoch zusätzlich ein High-Performance Echtzeitsystem und vor allem eine Kopplung der einzelnen Prozessoren (also tatsächlich ein verteiltes Echtzeitsystem) notwendig. Daher sind im zweiten Jahr des Projekts zusätzliche Förderungsmittel dafür notwendig. *Dauer: 5-7 Monate.*

3. Angaben zur Forschungsstätte und zu Förderungsmitteln

3.1 Ort der Forschung

Das beantragte Forschungsprojekt soll an der *TU-Wien*, und zwar am *Institut für Technische Informatik*, Abteilung für *Automatisierungssysteme* (*Leiter: O.Prof. Dr. G.-H. Schildt*) durchgeführt werden. Die Abteilung (1 Professor, 4 Univ. Assistenten, 1 Techniker, 1 Sekretärin) wurde 1988 gegründet und befindet sich in der *Treitlstraße 3, 1040 Wien*.

Wie schon erwähnt, liegt das Projekt genau in einer der primären Forschungsrichtungen der Abteilung; sowohl der Antragsteller als auch einer der direkten Projektmitarbeiter sind dort als Univ. Ass. beschäftigt. Dadurch können die umfangreichen Ressourcen der Forschungsstätte sehr intensiv genutzt werden. Dies betrifft unter anderem

- *Workstations*
  Insgesamt können max. vier(!) der an der Forschungsstelle vorhandenen HP/Apollo-Workstations für das Projekt verwendet werden.

- *Personal*
  Dies betrifft neben den primären Projektmitarbeitern v.a. eine Hilfestellung durch die anderen Univ. Ass. und durch den Techniker.
  An dieser Stelle sind natürlich auch die (an Studenten zu vergebenden) Informatik-Praktika zu erwähnen, an die u.U. einfachere Implementierungstätigkeiten delegiert werden können.

- *Räumlichkeiten*
  Für alle Mitarbeiter des Projektes stehen geeignete Räumlichkeiten zur Verfügung.

- *Sonstiges*
  Ebenfalls zu erwähnen sind natürlich die Möglichkeiten zur Benutzung der Instituts- und der TU-Bibliothek. Nicht unwichtig ist auch die Ausnutzung der bestehenden Kontakte zu anderen Forschungsstellen, etwa anderen TU-Instituten oder außeruniversitären Einrichtungen wie dem Forschungszentrum Seibersdorf, der TU Braunschweig oder INRIA Rocquencourt.

3.2 Personalressourcen

Dieser Abschnitt enthält eine Aufstellung der Mitarbeiter an dem zuvor beschriebenen Forschungsprojekt VTA. Betreffend die einzelnen Aufgabengebiete siehe auch Abschnitt *Projektdurchführung* (Seite 2-8) bzw. *Wissenschaftliche Fragestellungen* (Seite 1-3).

3.2.1 Vorhandenes Personal

Als direkte Mitarbeiter am Projekt VTA sind zwei an der Forschungsstätte vollbeschäftigte Univ. Ass. zu nominieren:

(1) **Univ. Ass. Dr. Ulrich Schmid (Antragsteller, Projektleiter)**
Die zentrale Aufgabe von Dr. Schmid ist die Organisation und Koordination des Gesamtprojektes. Dies umfaßt unter anderem die Ausarbeitung einer globalen Basis–Spezifikation für den VTA und in weiterer Folge die Abstimmung der von den einzelnen Mitarbeitern geleisteten Arbeit. In diesem Zusammenhang ist auch die Betreuung der Diplomarbeiten der Projektmitarbeiter *Wolfgang Kastner*, *Thomas Hontsch* und *Günter Glaser* zu erwähnen.

Wie schon bemerkt, ist das vorliegende Projekt nicht zuletzt für die Forschungstätigkeit des Antragstellers sehr wichtig. Deshalb ist die Beantwortung der wissenschaftlichen Fragestellungen (3), vor allem in Hinblick auf den den mathematischen Hintergrund, als ganz wesentliche Aufgabe von Dr. Schmid anzusehen. Hierbei ist eine nicht unbeträchtliche Unterstützung durch einen weiteren, an der Forschungsstätte beschäftigten Univ. Ass., **Dr. Johann Blieberger**, zu erwarten.

(2) **Univ. Ass. DI Stefan Stöckler**
Die Aufgabe von DI Stöckler ist, grob gesagt, die Beantwortung der wissenschaftlichen Fragestellungen (2), vor allem betreffend den Problemkreis (e); diese Arbeit soll letztlich eine Dissertation ergeben. Darüberhinaus obliegt ihm die Unterstützung bzw. Betreuung der Diplomarbeit von Hrn. *Johann Klasek*.

Ebenfalls hier zu erwähnen ist die Bereitschaft des Leiters der Forschungsstätte, O.Prof. Dr. G.–H. Schildt, im Falle des unvorhersehbaren Ausscheidens von Dr. Schmid die Agenden des Projektleiters (zusammen mit DI Stöckler) zu übernehmen.

3.2.2 Beantragtes Personal

Aus Förderungsmitteln wären insgesamt vier Diplomanden zu beschäftigen. Die bisherigen Erfahrungen lassen erwarten, daß die im folgenden nominierten Mitarbeiter ihre Aufgaben mit überdurchschnittlichem Erfolg erledigen werden. Da dies jedoch einen im Vergleich mit anderen Diplomarbeiten sowohl vom Schwierigkeitsgrad als auch vom Zeitaufwand überdurchschnittlichen Arbeitseinsatz erfordert, erscheint in allen Fällen eine *Forschungsbeihilfe bis zur Graduation* in maximaler Höhe (dzt. öS 10.000,––/Monat) angemessen.

Dies erscheint auch insoferne motivationsfördernd und somit eine ganz wesentliche Voraussetzung für das Gelingen des Projekts zu sein, als die für eine Diplomarbeit verhältnismäßig lange Dauer einen frühen Eintritt der durchwegs sehr hochqualifizierten Diplomanden in das (extrem gut bezahlte!) Berufsleben verhindert!

(3) **Wolfgang Kastner**
Die Aufgabe von Hrn. Kastner ist die Abdeckung des Teilgebietes (1), also der Bereitstellung eines Ethernet–basierenden, verteilten pSOS+m Multiprozessorsystems mit Clock–Synchronisation (siehe Anschnitt *Projektdurchführung*, Seite 2–8). Diese Tätigkeit sollte innerhalb *eines Jahres* mit der Diplomarbeit (Betreuer: Dr. Schmid) abgeschlossen werden.

Beschäftigung auf Basis einer Forschungsbeihilfe bis zur Graduation in maximaler Höhe.

(4) **Johann Klasek**
Die Aufgabe von Hrn. Klasek ist die Abdeckung des Teilgebietes (2), also der praktische Realisierung der VTA–Targets. Bei gewissen Teilaspekten ist eine direkte Zusammenarbeit mit DI Stöckler erforderlich. Da die projektierte Gesamtdauer die für eine Diplomarbeit zumutbare Dauer bei weitem übersteigt, ist hier folgende Vorgangsweise geplant:

- *1. Jahr*: Mitarbeit an den wissenschaftlichen Fragestellungen bzw. Entwicklung von darauf aufbauenden Systemkonzepten auf der Basis einer Forschungsbeihilfe bis zur Graduation in maximaler Höhe; Abschluß mit der Diplomarbeit (Betreuer: DI Stöckler).

- *2. Jahr:* Erledigung der verbleibenden Arbeiten auf der Basis eines Dienstvertrages (vollbeschäftigter Vertragsassistent).

(5) **Thomas Hontsch**
Hrn. Hontsch obliegt die praktische Arbeit betreffend das Teilgebiet (3), also die Erstellung der Low–Level VTA–Host Software. Da auch hier die projektierte Gesamtdauer den für eine Diplomarbeit zumutbaren Rahmen bei weitem übersteigt, ist dieselbe Vorgangsweise wie vorher geplant:

- *1. Jahr*: Arbeit an den wissenschaftlichen Fragestellungen bzw. an den daraus resultierenden Konzepten auf der Basis einer Forschungsbeihilfe bis zur Graduation in maximaler Höhe; Abschluß mit der Diplomarbeit (Betreuer: Dr. Schmid).

- *2. Jahr:* Erledigung der verbleibenden Arbeiten auf der Basis eines Dienstvertrages (vollbeschäftigter Vertragsassistent).

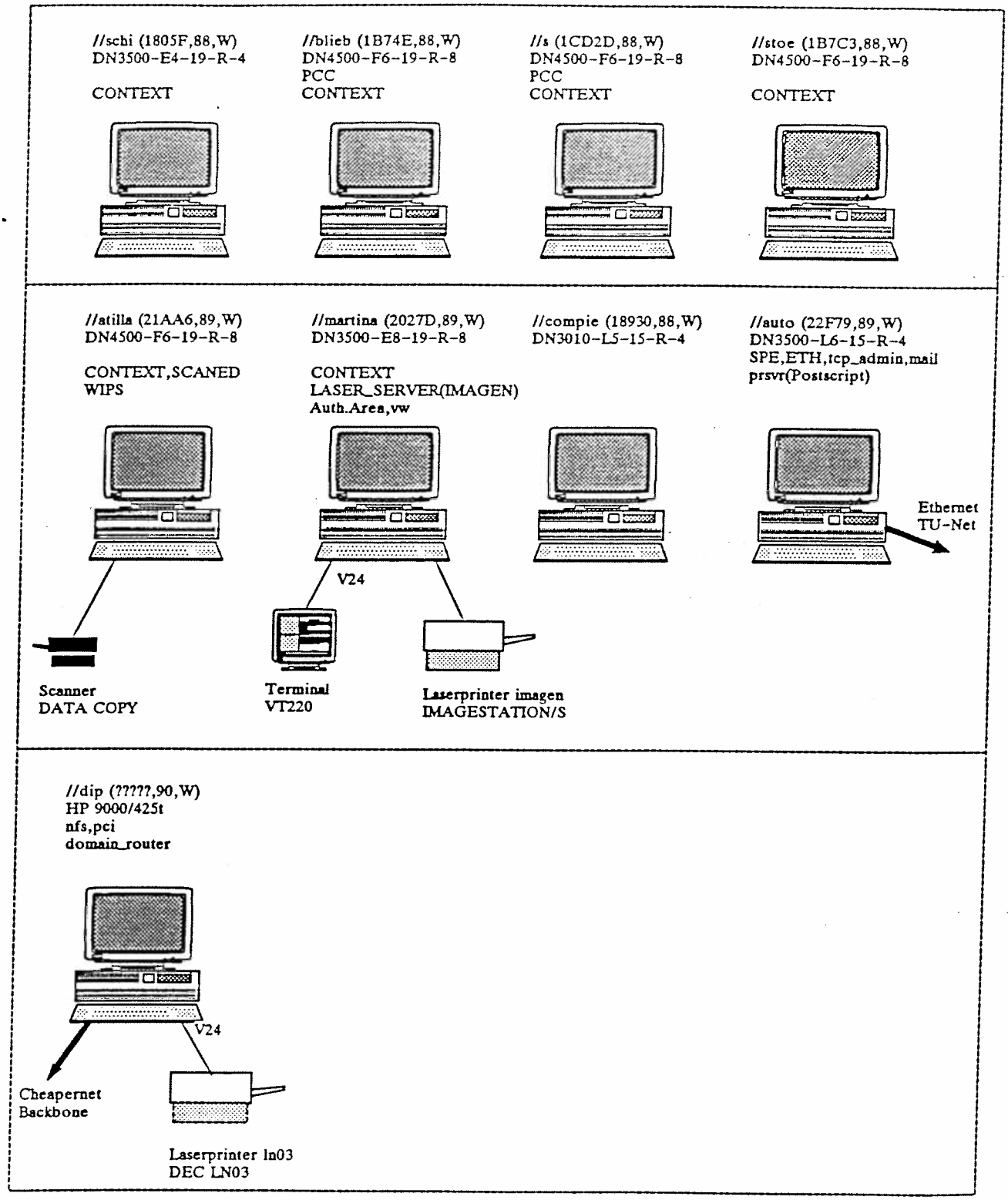
(6) **Günter Glaser**
Hrn. Glaser obliegt die praktische Arbeit betreffend das Teilgebiet (4), also die Erstellung der High–Level VTA–Host Software. Hierbei ist eine enge Zusammenarbeit mit Dr. Schmid erforderlich. Diese Tätigkeit sollte innerhalb *eines Jahres* mit der Diplomarbeit (Betreuer: Dr. Schmid) abgeschlossen werden.

Beschäftigung auf Basis einer Forschungsbeihilfe bis zur Graduation in maximaler Höhe.

3.3 Vorhandene Sachausstattung

Die Forschungsstätte verfügt über eine sehr umfangreiche Geräteausstattung mit modernsten HP/Apollo-Workstations. Konkret handelt es sich dabei um zwei autarke Token-Ring Networks, den die Workstations der Abteilungsmitglieder verbindenden *Instituts-Ring* und den die Workstations und PCs für die Übungen zur "Prozeßautomatisierung" verbindenden *Übungsring*. Hierzu ist zu bemerken, daß zu den Lehr-Aufgaben der Forschungsstätte die Organisation der Vorlesung und der Laborübung "Prozeßautomatisierung" gehört, die als Pflichtlehrveranstaltung im 4. Semester von ca. 300-400(!) Informatikern besucht wird.

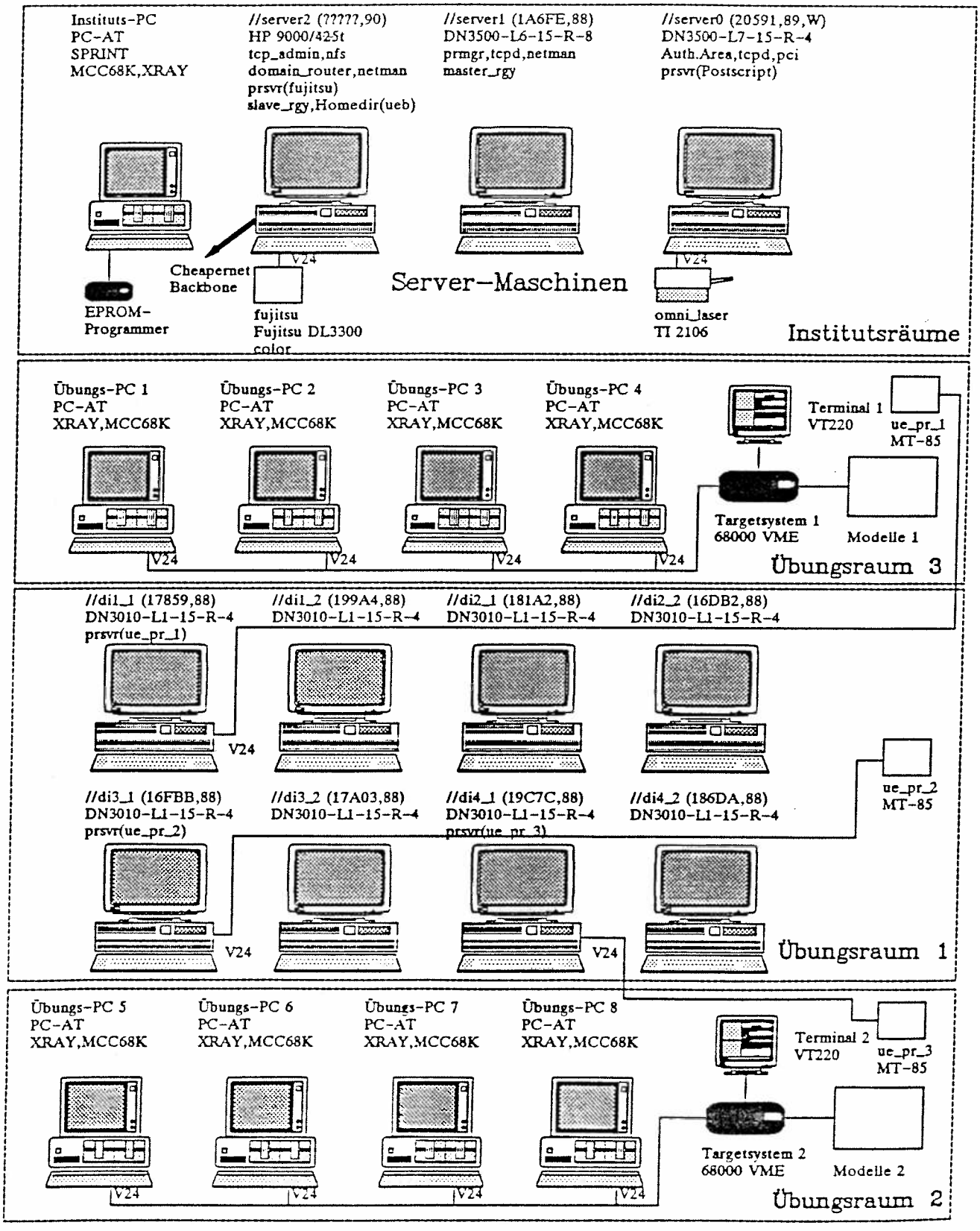
Das folgende Bild zeigt die Maschinenkonfiguration des Instituts-Ringes; bei den mit DN (etwa DN4500-F6-19-R8) bezeichneten Maschinen handelt es sich um Apollo-Workstations, bei den HP 9000 Systemen um die Nachfolgermodelle der Firma HP (hierbei ist zu beachten, daß die Firma Apollo von HP übernommen wurde). Fast alle diese Maschinen sind mit 350 MB Disks und hochauflösenden 19" Farbmonitoren ausgerüstet.



Maschinenkonfiguration Institutsring

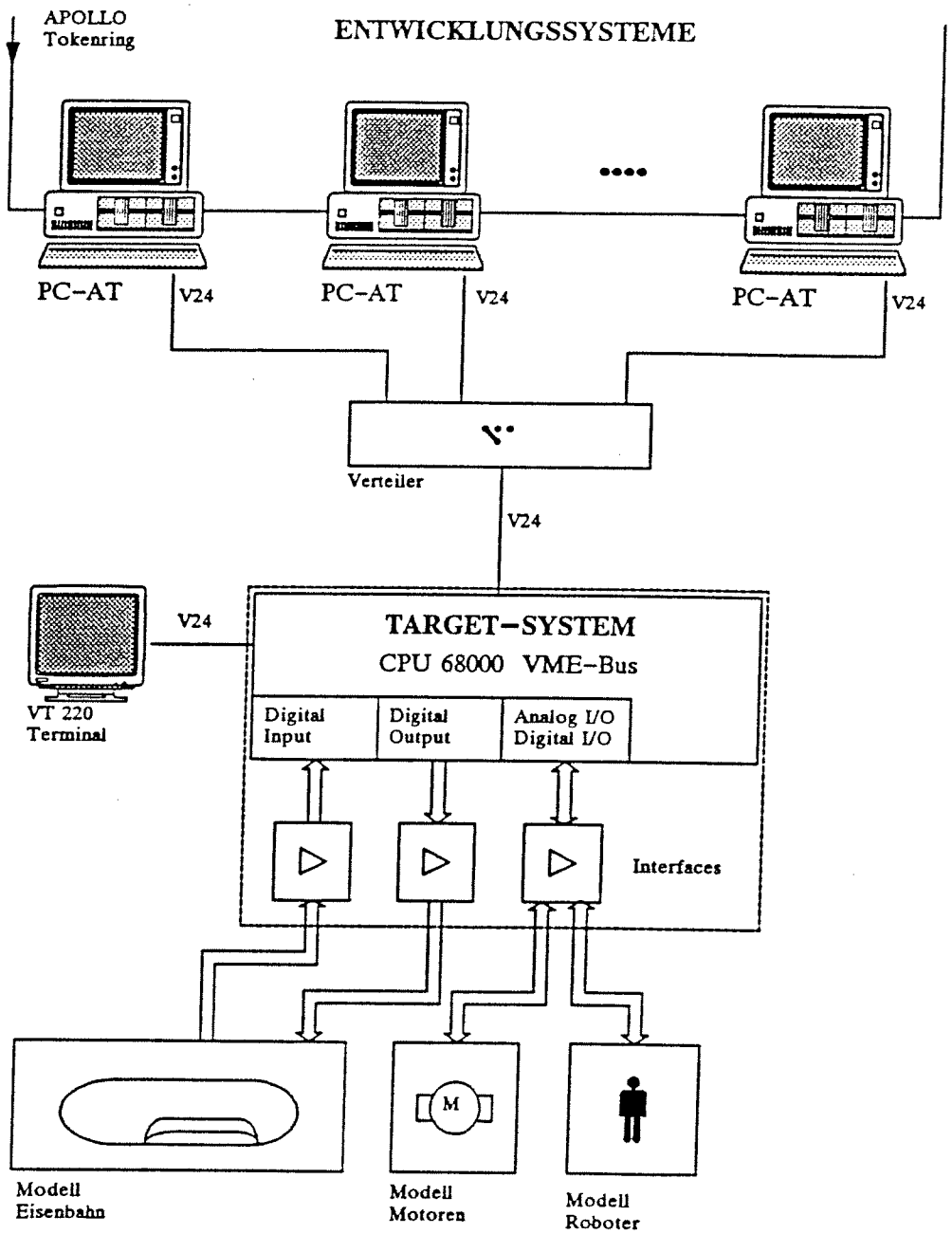
Die obigen Maschinen stehen hauptsächlich in den Zimmern der einzelnen Angehörigen der Forschungsstätte. Für das Projekt VTA können davon die (Color-)Workstations von Dr. Schmid (//s), DI Stöckler (//stoe) und die Maschine im Diplomandenraum (//dip) verwendet werden.

Das folgende Bild zeigt die Konfiguration des hauptsächlich für die jeweils im Sommersemester stattfindenden Übungen zur "Prozeßautomatisierung" und Informatik-Praktika vorgesehenen Übungs-Ringes:



Maschinenkonfiguration Übungsring

Neben drei Server-Maschinen, von denen eine Color-Workstation (//server2, leider nur im Wintersemester verfügbar) für das Projekt bereitsteht, befinden sich nur diskless Workstations und PC-ATs im Übungs-Ring. Zu erwähnen sind allerdings die beiden 68000 VME-Targetsysteme, die (ebenfalls im Wintersemester) für Testzwecke bereitstehen. Deren Aufbau (und Anschluß an die PCs) ist folgendem Bild zu entnehmen:



Aufbau der Targetsysteme für die Übungen zur "Prozeßautomatisierung"

Der Instituts- und der Übungsring sind durch ein Backbone-Netzwerk auf der Basis von Ethernet (Thin Wire) verbunden; hier sollte auch die aus Projektmitteln

anzuschaffende Sún–Workstation und die pSOS+m – Systeme angeschlossen werden. Im Endeffekt wäre dieses Ethernet–Backbone daher die für den VTA notwendige Cheapernet–Verbindung von VTA–Host und VTA–Targets (siehe Abschnitt *Projektziele*, Seite 2–1).

Für das Projekt nötige Software ist teilweise bereits vorhanden. So existiert etwa die für die Integration der aus Projektmitteln anzuschaffenden Sun–Workstation erforderliche TCP/IP– und NFS–Software sowie das X–Windows und MOTIF für die Apollo–Maschinen.

Ebenfalls bereits vorhanden sind wesentliche Teile der pSOS–Systemsoftware. Dabei handelt es sich um pSOS+ (Single Processor 68000 Kernel), pROBE+ (System State Debugger) und pREPC+ (C–Interface).

3.4 Beantragte Geräte, Systemkomponenten und Software

Die im folgenden aufgeführten Komponenten sind für die Durchführung des gegenständlichen Projektes notwendig und werden selbstverständlich ausschließlich dafür eingesetzt. Sie wären zur Gänze aus Projektmitteln zu finanzieren.

In Anbetracht der zuvor beschriebenen Ausstattung der Forschungsstelle ist es offensichtlich, daß es sich bei den benötigten Geräten nicht um Teile der infrastrukturellen Grundausstattung handeln kann. Es ist vielmehr so, daß das Projekt VTA sehr ausgiebig die vorhandene Infrastruktur der Forschungsstelle (4 Workstations, Software, Räumlichkeiten, ...) nutzen kann. Bei den zusätzlich benötigten Geräten handelt es größtenteils um ganz spezielle Dinge, die an Nachbarinstituten nicht vorhanden sind; eine Mitbenutzung kommt daher nicht in Frage.

Da das Projekt, wie schon erwähnt, in der Forschungsrichtung der Forschungsstätte liegt, ist es darüberhinaus klar, daß *Verbrauchsmaterial*, *Publikationskosten* usw. über die ordentliche Dotation der Abteilung abgerechnet werden können. Nicht möglich ist allerdings die Verwendung von Mitteln aus der außerordentlichen Dotation. Dies hat hauptsächlich folgende Gründe:

- Es gibt eine (Fachgruppen–interne) Konvention, daß Ordinariate, die Berufungsmittel ausständig haben, keinen Anspruch auf Mittel aus der a.o. Dotation haben. Die Berufungsmittel von O.Prof. Schildt waren 1989 aufgebraucht, die daraufhin eingebrachten Anträge wurden jedoch durch eine von BMWF durchgeführte Umorganisation der EDV–Ausrüstung für die Lehre ("Diebold–Studie") verzögert: Anstelle des dringend benötigten Ausbaus der Infrastruktur des Instituts–Ringes wurden zwei Server–Maschinen für den Übungsring bewilligt, die (hoffentlich) Ende November 1990(!) eintreffen werden (eine davon, //server2, soll für das Projekt mitverwendet werden). Aus diesem Grunde hat die Abteilung 182/3 bis jetzt noch *keinerlei Forschungs–Mittel aus der a.o. Dotation* erhalten.

- Die frühestens 1991 zu erwartenden a.o. Mittel für die Forschung, deren Höhe sich durch die Umorganisation der EDV–Ausrüstung für die Lehre empfindlich verringert wird, werden ganz sicher allein von dem vorhin

erwähnten infrastrukturellen Ausbau aufgebraucht (dabei handelt es sich vor allem um benötigte Serverkapazität); ganz zu schweigen von der Tatsache, daß andere (kleinere) Forschungsprojekte der Forschungsstätte ebenfalls finanziert werden müssen.

Konkret handelt es sich bei den aus Förderungsmitteln zu finanzierenden Geräten um folgende Einzelkomponenten (siehe dazu auch Abschnitt *Projektdurchführung*, Seite 2-6):

(1) **Sun-Workstation**, Angebot A, Preis: öS 176.552,-- excl. Mwst.

Hierbei handelt es sich um eine Sun SPARCstation IPC mit einem Thin Wire Ethernet-Anschluß, einem externen Magnetbandkassettenlaufwerk, dem Betriebssystem SunOS 4.1 und der kompletten Systemdokumentation.

Die Notwendigkeit für die Anschaffung der Sun-Workstation ergibt sich primär aus der Tatsache, daß der für die Entwicklung einer Software für pSOS-Systeme unbedingt notwendige High-Level Debugger XRAY+ für die an der Forschungsstätte vorhandenen Workstations nicht erhältlich ist (und es lt. Auskunft des Distributors auch nie sein wird).

Die hinter dieser und vergleichbaren Entwicklungsumgebungen stehende Philosophie geht von einem getrennten Entwicklungs- und Zielsystem aus. Alle Phasen der Software-Entwicklung (etwa Editieren, Compilieren) werden auf einer leistungsfähigen "kommerziellen" Entwicklungs-Maschine (einer Workstation) erledigt. Die entwickelten Programme werden dann mit Hilfe des Debuggers (XRAY+) in das Zielsystem geladen und können (vom Entwicklungssystem aus bedient!) so getestet werden, als würden sie lokal am Entwicklungssystem laufen (obwohl sie in Wirklichkeit im Zielsystem exekutiert werden). Die Verbindung zwischen dem Entwicklungs- und dem Zielsystem kann im einfachsten Fall über eine serielle V24-Leitung oder aber über Ethernet oder gar einen In-Circuit - Emulator erfolgen.

Wie schon erwähnt, wird bei den Übungen zur "Prozeßautomatisierung" die PC-Version des XRAYs eingesetzt. Diese ist jedoch aus zwei Gründen für die Abwicklung des Projektes völlig unbrauchbar:

- *Verbindung über eine V24-Schnittstelle*
  Die Kopplung zwischen dem Entwicklungs- und dem Zielsystem erfolgt über eine serielle (Punkt-zu-Punkt) Leitung, die mit max. 19200 Bd betrieben werden kann. Dadurch wird das Downloading selbst sehr kurzer Programme unzumutbar langsam; ein ca. 100-zeiliges C-Programm benötigt etwa 30-60 Sekunden (je nach der Anzahl der benötigten Library-Funktionen). In Anbetracht der sicherlich äußerst umfangreichen Software für die VTA-Targets (einige zehntausend Zeilen sind sicher realistisch) ist das absolut unzureichend.

Ein ebenfalls sehr unangenehmer Nachteil ist die Tatsache der Punkt-zu-Punkt Verbindung. Da es sich bei den VTA-Targets um ein verteiltes System handelt, also mehrere pSOS-Systeme simultan zu debuggen sind, müßte für jedes Target ein eigener Entwicklungs-PC bereitgestellt werden; dies stellt eine im Zeitalter der Multi-Window Workstations äußerst unübliche gewordene Vorgangsweise dar.

- *Limitierung der Funktionalität*
  Die PC-Variante der Entwicklungsumgebung ist eine "abgemagerte" Version der Workstation-Variante. Dies betrifft nicht nur das Multi Windowing (und damit den Multiprocessor-Support), sondern vor allem die maximale Größe der entwickelbaren Programme; hier schafft auch ein Extended Memory o.ä. keine Abhilfe. Dieser Punkt betrifft übrigens nicht nur den XRAY, sondern auch den C-Compiler MCC68K; mehr als ca. 300-zeilige C-Programme sind nicht zu übersetzen.

Aus diesen Gründen ist die Anschaffung der Workstation-Variante des XRAY+ nicht zu umgehen; diese ist aber leider nur für Sun-Maschinen erhältlich.

Mit der erwähnten Sun-Workstation steht aber nicht nur die notwendige Hardware-Plattform für die Entwicklungsumgebung für pSOS-Systeme, sondern auch gleich die benötigte, leistungsfähige(!) Hardware für den VTA-Host zur Verfügung! Durch die Kompatibilität der UNIX-Betriebssysteme (X-Windows) bzw. des MOTIF auf Sun- und Apollo-Maschinen kann die Software für den VTA-Host aber ohne weiteres auf den Apollos entwickelt werden; für die endgültige Portierung auf die Sun sollte lediglich eine Neucompilierung notwendig sein.

Das externe Magnetbandkassettenlaufwerk ist notwendig, um die auf der Workstation zu installierende Software (inklusive SunOS ca. 100-150 MBytes) auf die Platte einlesen zu können; die Integration in das Apollo-Netzwerk erlaubt leider die Verwendung der dort befindlichen Bandstationen nicht.

Zum Schluß sollte nicht unerwähnt bleiben, daß (nach den Erfahrungen bei der Anschaffung der Geräte für die Forschungsstätte) der Gesamtpreis ausgesprochen günstig ist. Da die Firma Bacher den Generalvertrieb für Sun in Österreich hat, ist übrigens kein Konkurrenzoffert verfügbar.

(2) **MOTIF für Sun-Workstation**, Angebot B, Preis: DM 1.600,-- excl. Mwst.

Die Standard-Oberfläche einer Sun basiert auf OPEN LOOK, welches nach den gegenwärtigen Trends am UNIX-Markt dem OSF/MOTIF wahrscheinlich nicht standhalten wird. Aus diesem Grunde soll die Oberfläche des VTAs auf MOTIF aufgesetzt werden.

(3) **MCC68K f. Sun-Workstation**, Angebot C, Preis: öS 74.120,-- excl. Mwst.

Dabei handelt es sich um den C-Crosscompiler sowie den zugehörigen Assembler, Linker und Librarian für die Entwicklung von pSOS-Software. Siehe dazu auch die Bemerkungen im Punkt (1).

(4) **XRAY+ f. Sun-Workstation**, Angebot C, Preis: US $ 3.315,-- excl. Mwst.

C High Level Language Debugger (Ethernet-Version) für die Entwicklung von Software für pSOS-Systeme. Diese Komponente ist nicht für Apollo-Workstations zu bekommen; siehe auch die Bemerkungen im Punkt (1).

(5) **INTERLEAF f. Sun, Apollo**, Angebot D, Preis: DM 18.530,-- excl. Mwst.

Die Durchführung des Projektes erfordert gerade in der ersten Phase (voraussichtlich Sommersemester 1990) ein leistungsfähiges verteiltes Document Management System mit einer ausreichenden Anzahl (günstig wären sechs) von gleichzeitig benutzbaren Arbeitsplätzen.

Wie im Abschnitt *Projektdurchführung*, Seite 2-8 näher erläutert erfolgt dabei die Bearbeitung der wissenschaftlichen Fragestellungen, und zwar mit dem Ziel einer globalen Spezifikation für den VTA. Diese Tätigkeit erfordert, daß alle Mitarbeiter des Projekts gleichzeitig und voneinander unabhängig ihre Teilgebiete bearbeiten und, als "einziges" Resultat, Dokumentation produzieren müssen. Aus dieser wird dann (im Zuge der Abstimmungsbesprechungen) ein einziges Gesamtdokument erstellt. Eine solche Vorgangsweise ist heutzutage bei Teamarbeiten in der Informatik gang und gäbe; die Zeit des Papier-und-Bleistift - Designs ist durch die enormen Fortschritte der Computer-Netzwerke und der darauf aufbauenden kommerziellen Software zum Glück vorbei.

An dieser Stelle erscheint es passend, eine vielleicht nicht überall bekannte (finanziell unangenehme) Eigenheit von Workstations zu erwähnen: Workstations haben eine firmenweit eindeutige, in der Hardware festgelegt "Seriennummer", die *Host-* oder *Node-ID* genannt wird. Diese Nummer wird von Software-Herstellern dazu benutzt, ihre Produkte *node-locked* zu machen, speziell bei teuren Produkten. Eine derartige Software läuft nur auf einer ganz bestimmten Maschine!

An der Forschungsstätte ist nun ein sehr leistungsfähiges (und entsprechend teures) Document Management System (6 Lizenzen CONTEXT von Mentor Graphics) im Einsatz, das selbstverständlich node-locked ist. Es wäre für das Projekt VTA nur auf den Workstations von Dr. Schmid und DI Stöckler vorhanden. Selbst die Beschaffung zweier weiterer Lizenzen für die Apollo-Maschinen //dip und //server2 würde die Anzahl der verfügbaren Arbeitsplätze nicht ausreichend erhöhen (es würden sich dadurch nur drei/vier verfügbare Arbeitsplätze ergeben, die Maschine //server2 ist ja im Sommersemester durch den Übungsbetrieb blockiert). Für die Sun-Workstation ist das CONTEXT gar nicht verfügbar.

Eine wesentlich bessere (und darüberhinaus auch bedeutend billigere) Alternative besteht darin, das University Program Support - Programm der Firma INTERLEAF aufzunutzen und zu einem wirklich extrem günstigen Preis ein Document Management System zu beschaffen, das in Punkto Leistungsfähigkeit beinahe an CONTEXT herankommt, geradezu als eine Art Standard angesehen werden kann und nicht zuletzt auf Sun und Apollo-Maschinen zu haben ist. Es sollten daher

- *1 UPS-Lizenz für eine Sun, 1 Datenträger mit der Sun-Software und 1 Satz Dokumentation*

- *4 UPS-Lizenzen für Apollos, 1 Datenträger mit der Apollo-Software und 1 Satz Dokumentation*

aus Projektmitteln angeschafft werden. Der Firma Interleaf in München obliegt der Generalvertrieb dieses Systems; ein Konkurrenzoffert ist daher nicht verfügbar.

(6) **2 identische VTA-Targets**, Angebot E, A, F
Preis: 2 x öS 80.747,50 = öS 161.495,--

Wie im Abschnitt *Projektziele* (Seite 2-1) näher erläutert erfolgt die eigentliche Ankopplung des VTAs an ein verteiltes Echtzeitsystem über die VTA-Targets. Bei der Hardware eines solchen VTA-Targets handelt es sich im Prinzip lediglich um eine VME CPU-Komponente der Firma Force, siehe dazu die diesbezüglichen Bemerkungen im Abschnitt *Projektdurchführung* (Seite 2-6). Darüberhinaus ist natürlich auch ein (Thin Wire) Transceiver für die Ankopplung an das Cheapernet und ein Rack (also Gehäuse, VME-Motherboard und Stromversorgung) notwendig. Dazu ist zu bemerken, daß es sich bei VME-Komponenten um einzelnen Boards (Printplatten) handelt, die für sich allein natürlich nicht funktionstüchtig sind.

Da eine ganz wesentliche Voraussetzung des Projektes das Monitoring verteilter Echtzeitsysteme ist, sind natürlich auch mehrere VTA-Targets notwendig; aus Kostengründen erscheint es jedoch sinnvoll, sich im ersten Jahr des Projekts sich zwei derartige Systeme zu beschränken.

Bei der Auswahl des CPU-Boards waren folgende Kriterien maßgeblich:

- Leistungsfähige CPU (68030) mit großem lokalen, dual-ported RAM, EPROM für die VTA-Software und Timer-Funktionen für die Clock-Synchronisation.

- Unterstützung des *Force Message Broadcasts* (FMB). Auf diese Weise kann eine sehr effiziente Erfassung der aus dem Echtzeitsystem kommenden Events realisiert werden, ohne eine eigene Spezial-Hardware entwickeln zu müssen, siehe auch die Bemerkungen im Abschnitt *Projekt-*

durchführung (Seite 2–6).

- On–Board Ethernet–Controller.

Diese Kriterien werden alle von der Force CPU–30 XBE erfüllt. Dieses Board wird darüberhinaus auch von der pNA+ Software und von der pSOS–Entwicklungsumgebung (XRAY+, pROBE+) unterstützt. Nötig ist dafür nur ein spezielles Boot–PROM (siehe Punkt (7)).

Zu den notwendigen Racks ist zu bemerken, daß gerade die Teilaufgabe (1) bereits zu Beginn des Projeks eine betriebsfähige Hardware voraussetzt, siehe dazu Abschnitt *Projektdurchführung* (Seite 2–8). Die Racks der für die Übungen zur "Prozeßautomatisierung" eingesetzten Target–Systeme sind jedoch im Sommersemester nicht verfügbar. Darüberhinaus befinden sie sich in verschiedenen Räumen (und außerdem sehr weit von den Institutsräumlichkeiten) entfernt, sodaß sie aus arbeitsorganisatorischen Überlegungen nicht in Frage kommen.

(7) **pSOS–Software**, Angebot E, Preis: US $ 6.090,––

Wie bereits im im vorigen Abschnitt erwähnt, ist ein Großteil der pSOS–Software bereits vorhanden. Notwendig ist lediglich:

- Upgrade der vorhandenen (Singleprocessor 68000) pSOS+ – Version auf das (Multiprocessor 68030) pSOS+m

- Anschaffung der pNA+ – Software. Hierbei handelt es sich um eine TCP/IP – Software für pSOS–Systeme, auf der die Kommunikation der (pSOS) VTA–Targets mit dem (UNIX) VTA–Host aufgebaut werden soll.

- Boot–PROM für die pNA+ – Software bzw. für die pSOS–Entwicklungsumgebung (XRAY+, pROBE+) für die Force CPU–30. Dieses ist notwendig, um die angesprochenen Software–Komponenten auf der CPU–30 verwenden zu können.

An dieser Stelle sollte noch einmal explizit darauf verwiesen werden, daß die meisten pSOS–Komponenten bereits an der Forschungsstätte existieren.

(8) **Zusätzliche VME–Hardware (erst im 2. Jahr)**, Angebot E
Preis: 2 x öS 55.291,50 = öS 110.583,––

Im zweiten Jahr des Projekts müßten für Testzwecke zwei weitere CPU–30 und zwei Thin Wire Transceiver bereitgestellt werden. Dieser Aufwand ist folgendermaßen zu begründen:

- Die bei den Übungen zur "Prozeßautomatisierung" eingesetzten Target–Systeme sind <u>nicht</u> untereinander gekoppelt, stellen also kein (typi-

sches) verteiltes Echtzeitsystem dar. Darüberhinaus sind die dort verwendeten Force CPU-6 (68000 Prozessor) nicht dazu geeignet, hohe Anforderungen (in Punkto Geschwindigkeit) an den VTA zu stellen. Es ist daher notwendig, die beiden CPU-6 Boards der existierenden Übungssysteme durch je eine CPU-30 zu ersetzen.

- In dieser späteren Phase des Projekts ist die Minimalkonfiguration mit nur zwei VTA-Targets nicht ausreichend, um Probleme im Zusammenhang mit der verteilten Event-Recognition und der Clock-Synchronisation zu entlarven. Es ist daher notwendig, ein etwas größeres Netzwerk an VTA-Targets aufzubauen und zu testen. Hierfür ist übrigens kein zusätzliches Rack erforderlich, da in dieser späten Phase des Tests die gesamte Hardwarekonfiguration des VTAs in die Übungsräume transportiert werden kann; aus diesem Grunde stehen die Racks der Übungssysteme zur Verfügung.

3.5 Beantragtes Material

Diese Kosten können aus der ordentlichen Dotation der Forschungsstelle gedeckt werden.

3.6 Reisekosten

Da in Punkto Reisekosten die ordentliche Dotation nicht verwendbar ist, sollten aus Förderungsmitteln zwei Reisen von je einer Person zum Zwecke der Präsentation der Projektergebnisse, v.a. in die USA, finanziert werden. Dafür sollten etwa öS 30.000,-- vorgesehen werden.

Die Tagungsbebühren können aber problemlos aus der ordentlichen Dotation der Forschungsstätte bezahlt werden.

3.7 Publikationskosten

Diese Kosten können aus der ordentlichen Dotation der Forschungsstelle gedeckt werden.

4. Allfällige zusätzliche Angaben

Es wurde bei keiner anderen Stelle um Subventionen nachgesucht.

Weiters ist es denkbar, zusammen mit einer noch zu findenden Firma und dem Forschungsförderungsfonds der Gerwerblichen Wirtschaft ein für die industrielle Praxis verwendbares Instrument (auf Basis des VTAs) zu entwickeln.

Betreffend die Verwertungsmöglichkeit der beantragten Geräte ist zu bemerken, daß sich gegenwärtig ein weiteres, zum Projekt VTA thematisch gewissermaßen "duales" Forschungsvorhaben (Real Time Simulation) im frühen Planungsstadium befindet, welches die beantragte Gerätekonfiguration praktisch zur Gänze wiederverwenden könnte.