1

P C A

(Portable Communication Architecture)

ESPRIT 2434

FINAL IMPLEMENTATION REPORT

P C A

(Portable Communication Architecture)

A Portable Communication Architecture for Mixed Software and Hardware Environments

Jörg - Ingo Jakob

Philips GmbH Forschungslaboratorien
Forschungsabteilung Technische Systeme Hamburg

1. Problem Specification And Motivation *and*
2. Requirements Specification

One aspect of CIM is integrating islands of automation. A typical obstacle encountered here is the incompatibility of hardware and/or software islands which hinders information exchange. A step towards integration would be the availability of a communication software module which would run on practically any hardware platform and operating system, where two or more such modules could set up a standardized communication protocol[1]. We are looking for a portable and language independent architecture for continuous exchange of messages in a heterogenous computer network. The standardized message exchange shall be file-based, it shall use local or wide area networks, it shall run automatically and repeatedly (under software control), and it shall obey real time constraints which are typical for factory environments [defined here as transmission times for short messages (<= 1 kByte) in the range of one second]. Furthermore we specify the architecture to be concise (using only few pages of source code), to be completely and portably written in a common high level programming language without recoursing to low-level routines, and to be written in a way which allows for simple translation from one high level programming language to another.

3. Organization Structure

The potential role of the Portable Communication Architecture and its relations to other parts of the factory is documented by a GRAI information grid (Figure 1). The cells of the framed area are candidates for use of PCA. All information centres may benefit from the use of PCA. Of course, PCA can also be used in non-manufacturing areas with similar problems.

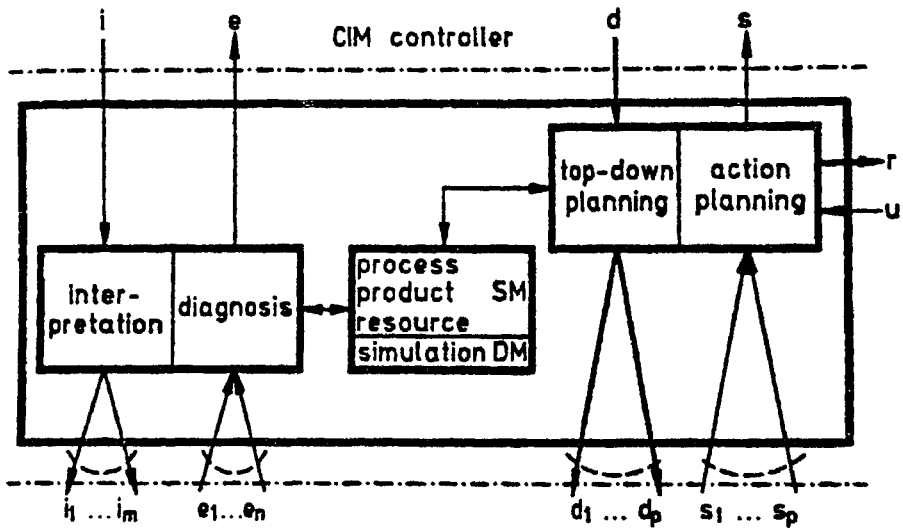
[1] KERMIT is an example of such a program, for use with serial links. KERMIT however is usually not available in source form, has a user interface not suitable for automated exchange of data, and does not make use of local (or wide) area networks. PCA has all these desirable features.

Horizon /Period	Function	Sales Management	Operational Management			Resource Management			Product Design
			Production	Quality	Maintenance	Material	Personnel	Equipment	
FACTORY	4 years 1 year		strategic plan *qty/prod.family* *start & end date*						
SHOP	1 year 1 month	order forecasts *qty/ prod.fam./ month*	master plan *qty/prod.type/ month*	quality master plan *products, equipment*	master plan for maintenance *maintenance procedures & availability, effort/w.station*	vendor capacities *qty/cpu/month* orders for common parts *qty/cpu/month*	forecasted manpower capacities *manpower/ qualification/ month*	equipment policy, machine layout *w.station spec., release dates*	development plan *prod.type spec., release dates*
	6 weeks 1 week	customer orders *qty/prod.type, customer priority*	production plan *qty/prod.type/ week*	quality information *quality level/ prod.type/week, quality level/ w.station/week*	maintenance plan *maint.orders/ w.station/week*	short term orders, external subassys *qty/cpu/week*	allocation of people *manpower/qual./ workcell/week*	installation & modification of equipment *installation plan/ w.station,date*	request for product modification *prod.type modification, release date*
	3 days 1 day		order dispatch list *qty/assembly, due date*	detailed quality analysis *quality level/ prod.type/day, quality level/ w.station/day*	released maintenance orders *maint.orders/ w.station/shift*	reserved parts & subassys *qty/cpu/week*	allocation of people to JIT or subassy production *manpower/qual./ workcell/ shift*	reservation of equipment *w.station/shift*	
WORK-CELL	3 shifts 1 hour		sequenced job list *job/w.station, start date*	quality data per w.station *quality level/ w.station*	machine status & maintenance actions *maint.orders/ w.station*	picking list *qty/part*	adjusted allocation *people/qual./ w.station*	tooling plan *tools/w.station shift*	alternative components list *qty/cpu/shift*

Figure 1: PCA and the information structure of a factory

4. Coordination Structure (CIM Controller)

As well as establishing communication between cells of the factory information grid, PCA can be used to establish communication between the constituting modules of the EP 2434 CIM Controller (Figure 2).



Models:

SM	static models
DM	dynamic model

Decision Flow:

d	decision frame from upper level
d_1-d_p	decision frame to lower level
r	request from/to same level

Information Flow:

e	status of resources to upper level
e_1-e_n	status of resources from lower level
i	status of upper level CIM module
i_1-i_m	status of this CIM module to lower level
s	decision frame status to upper level
s_1-s_p	decision frame status from lower level
u	request status from/to same level

Figure 2: EP 2434 Generic CIM Controller

5. Decision Processes and Decision Nets

Not applicable to this standard.

6. Representation and AI Modules

PCA is mainly routing files (messages), so it basically consists of program code for file routing, and of the internal representation of file names and routes (path names). File names and path names are generally stored as strings. Due to portability considerations, PCA is not designed as an object-oriented system. However, TUHH (Technische Universität Hamburg-Harburg) supplies a Smalltalk/V-based version of PCA.

7. Models

Not applicable.

8. Inference

Not applicable.

9. User Interface Guidelines

PCA has no physically visible user interface. The description of its software interface is to be found under Chapter 13.

10. Hardware And Software Requirements

Until now, PCA is available for and was tested on the following combinations of hardware and software:

Table 1: Successful communications implemented by the PCA. Numbers 1, 2, 3 indicate number of physical computers involved in test communication (cf. Figure 3). A double dash indicates an infeasible communication.

	Computer B			
	GENERA	MS-DOS	VAX/VMS	WINDOWS
Computer A				
GENERA 7.2	1, 2	3	2, 3	3
MS-DOS 3.3	--		2, 3	--
VAX/VMS 5.2			1, 2, 3	2, 3
WINDOWS 3.0				1

Being a portable standard, there are no known limitations on the range of hardware and software on which PCA could possibly be implemented.

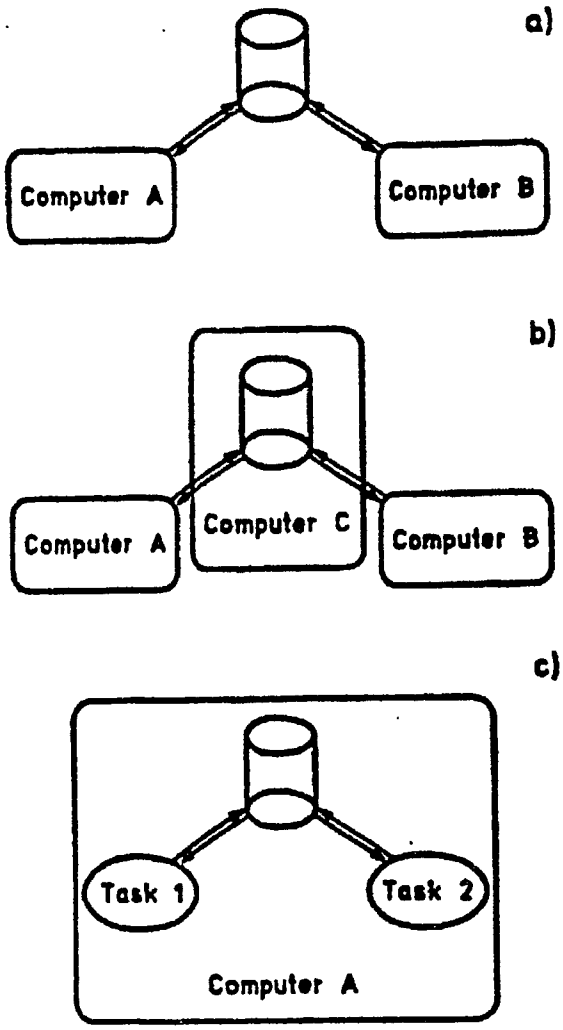


Figure 3: Typical number and arrangement of computers involved in a PCA-based communication

11. Prototype Implementation Experiences

PCA was implemented in ISO Pascal, ANSI, Common Lisp and Smalltalk/V so far. Each of these implementations can communicate with each other implementation, they can be used interchangeably.

PCA was tested on a variety of software and hardware combinations. The test consisted of sending a short message (an integer in ASCII representation) from a computer called "A" to a computer called "B" and backwards. During the test, each computer sends and receives 50 such messages without interruption. The computers A, B, C are always of the same brand and use the same operating system. Otherwise this measurement is similar to Measurement 1. 50 messages are sent. This measurement typically measures the sum of synchronization times of the Ethernet, of transmission times via Ethernet, and the time it takes to write/read onto the hard disk. Other times can be disregarded. Therefore, it should not matter

whether these measurements were performed using a C, Pascal or Lisp based version of PCA. Results:

No Name 386/33 PC-AT, PC-DOS 3.3, Turbo C++, Novell Netware 3.0:	0'03"
No Name 386/33 PC-AT, Windows 3.0/PC-DOS 3.3, Turbo C++, Novell Netware 3.0:	0'16"
VAXStation 3500 to VAXStation 3500, VAX/VMS 4.7/5.2, VAX Pascal 3.6:	2'21"
Symbolics 3640 to 3645, Genera 7.2, Symbolics Zetalisp:	3'25"
VAXStation 3500 to VAXStation 3500, VAX/VMS 4.7/5.2, VAXLisp 2.2:	7'58".

12. Software Module Structure

Due to its simpleness, PCA consists of one software module only (the main module).

13. Software Modules Interface Description

As mentioned in Chapter 12, PCA consists of one module only. However, two or more computers are communicating in any practical system, so two or more functionally identical PCA modules are involved in any such communication. This can be depicted as two identical, interacting state machines (Figure 4). The four states of each machine can be directly mapped onto functions or procedures in a high level programming language (see below).

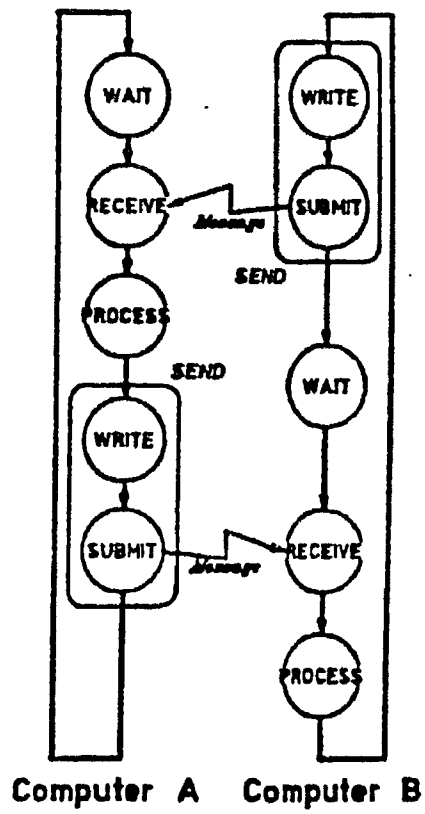


Figure 4: PCA depicted as two interacting state machines

The Common Lisp implementation of PCA will be used for the software module interface description, because it turns out to be the most concise and readable PCA.
The states of Figure 4 were implemented as follows (*Table 2*):

Table 2: Correspondence of PCA states and Lisp function names chosen

STATE	FUNCTION NAME
Wait	WaitForMessage(FileName)
Receive	ReceiveMessage(InFile)
Process	ProcessMessage(InFile)
Write	WriteMessage(TempFile)
Submit	SubmitMessage(TheFile OldFileName NewFileName)

All function parameters ending with -Name are of Common Lisp data type string. All function parameters ending with -File are of Common Lisp data type stream. The Common Lisp source is to be found in [Jakob 1991].

14. Training Manual

There is no Training Manual available. In [Jakob 1991], the PCA standard, its software structure and the PCA performance test mentioned in Chapter 11 are described in detail and with a commented PCA source code sample listing (Common Lisp version). Particularly, the performance test described may serve as an introduction to PCA.

15. References

[Jakob 1991] A Portable, Hardware and Language Independent Architecture for Communication (PCA); in: ESPRIT 2434 Consortium / Philips GmbH, ESPRIT 2434 30 Months Report, Task 2.5, Hamburg