Distributed Industrial Information Systems: Design and Experience

Paulo Veríssimo, Sérgio Melro, António Casimiro, Luís Silva
U. Lisboa - T. U. Lisboa - INESC
Department of Informatics,Bloco C5, Campo Grande, 1700 Lisboa - Portugal. Tel: +(351) 1 750 01 03, Fax: +(351) 1 757 78 31,
Email: pjv@di.fc.ul.pt, smm@inesc.pt, casim@inesc.pt, lgs@inesc.pt

Abstract

Distributed systems have proliferated in informatic applications during the last few years. Reasons for distribution are various, and come from different fields, such as: user requirements; technology; and market evolution itself.
The introduction of computers in office/factory floors is many often bottom-up. When the time comes to bridge the gap towards integration, the situation is sometimes chaotic, with a number of proprietary solutions, de-facto standards and customised protocols at stake.
The purpose of this paper is to discuss the motivations, design of, and real-life experience with, a *distributed industrial information system platform*, integrating the whole information flow of industrial facilities.

Keywords

CIM, SCADA, Distribution, Communication

1 INTRODUCTION

Distributed systems have proliferated in informatic applications during the last few years [PCB+91, Bee89, KBS91, JN90, KDK+89, McMS92, BGG+91, Coh87, JW86, Com91, ISA85]. Reasons for distribution are various, and come from different fields, such as: user requirements; technology; and market evolution itself. User applications requiring distribution per natura are increasing. Examples are cooperative applications intra- and inter organisations, and automation applications. Technology has its contribution, in the sense that it provides increasingly more effective networking facilities[PFB87, But86]. The market evolution must be taken into account, given its contribution to the widespread use of computers. This is true of any setting, from the individual user to corporate informatics parks.

It is known that the introduction of computers in companies, and more specifically in the office/factory floors, starts by the informatisation of individual machines (in the office or factory), and then groups of machines, and so forth. When the time comes to bridge the gap towards integration, the situation is sometimes chaotic, with a number of proprietary solutions, de-facto standards and customised protocols at stake. A stratification of the problems and standards for those layers has been claimed to be one solution to integration[Zim80].

Finally, a determinant factor leading the choice of system components when integrating industrial information systems is their cost. Unfortunately, it is easy to verify that the overall cost of an industrial information system tends to rise in the presence of heterogeneity, distributivity or expansibility requirements. It is then of fundamental importance to find new solutions which are capable of handling those requirements without additional costs.

The purpose of this paper is to discuss the design of a *distributed industrial information system platform*. We propose an architecture which integrates the whole information flow of industrial facilities: bottom-up (supervision) and top-down (command/control). We begin by sharing with the reader the user-oriented motivations of the options taken. Then, we present our architecture, and describe the design of the NavCim platform. Finally, we report real-life experience with the platform.

2 MOTIVATIONS

Industrial applications are distributed by nature. Inherent geographic separation and concurrent operation are two reasons. Surprisingly enough, it is the field where distribution has penetrated the least. To this situation certainly contributes the fact of the greater demands for dependability and timeliness vis-a-vis the lack of mature technology for genuinely decentralised operation[Ver93]. A control engineer would be upset if proposed to control a robot arm and a 1 ton press with two totally autonomous controllers merely exchanging messages without any supervisor (imagining the press going down with the arm under it...).

Still, a lot has been done in the direction of distribution, namely, the penetration of basic communication and file transfer protocols in current industrial systems[Com88, FFH+90, PJ85], and the inception of the MAP (Manufacturing Application Protocol) and CNMA (Communication Networks for Manufacturing Applications) standards[MAP85, ISO90b, CNM93]. Our claim is that the use of distributed technologies in an industrial setting is not only inevitable but also desirable, if mistakes of a recent past, namely within standardisation efforts, are not repeated.

Standards

Field buses, although the standardisation efforts have stalled for the last years, have contributed to cable sparing. In fact, they have been, and will largely remain, a sort of digital system over a wire, mainly with polled operation, to read sensors and drive actuators in a tightly synchronised manner[PFB87, ISA85, Int88]. Process buses, normally LANs, and upper tier backbones, have interconnected the company machines, mainly responding to information exchange and sharing needs (ISO 802.3, 802.4, FDDI). This is the scope of MAP/TOP and CNMA, which aim at interconnecting machines in the office and factory floors, allowing up- and download of programs, file transfer, inter-machine communication, etc[MAP85]. There have more recently attempts to bring the effort further down the road of distribution, by incorporating emerging standards such as ODP and DCE[CNM93, ODP87, Fou90]. In fact, as said before, distributed computing evolved much faster in the non-industrial world, where de-facto standards have not waited for ISO standardisation. These have occupied a market position which cannot be ignored, with the risk of insuccess or failure, which has already been encountered by some MAP component manufacturers. Examples are: UNIX "world", TCP/IP, NFS and AFS, widespread DBMS systems, RPC systems and OSF DCE, the ODP reference model, CORBA and other emerging object-oriented technologies. An engineering approach of today for factory integration must, if bound for success, bear in mind two crux factors: de-jure standards or what will be; de-facto standards or what there is.

SMEs

Another factor of weight when introducing automation and informatics is the SME scenario. SME needs in automation seldom call for a fully-fledged distributed architecture, neither do they call for large-scale solutions. With the advent of networking standards for automation, the problem has probably been seen "upside-down", and that may be a reason why SMEs have difficulty in investing in heavyweight solutions such as MAP or CNMA[GAT92]. It appears to us that the correct reasoning should be the following: a client looking for a system for a small solution with little distribution, should not pay the price for large-scale and intensive distribution. In consequence, when talking about automation, we should probably be talking about *distributable* and *scalable* systems, which are at the start centralised and small-scale. This is preferable to building heavy-artillery distributed and large-scale systems, which we then try to squeeze down and eventually re-centralise.

Non-Functional attributes

Whatever the problem to be solved, solutions are sometimes cluttered by an obvious characteristic of systems, but one often disregarded: the non-functional properties of the systems being built, and of the environments they control. As an example, it is of no use to define a very elegant object-oriented platform for computer control, to be driven transparently from any machine in the system, if it is known that the standard stack to be used, when operated for example from an office floor machine, will provide a round-trip delay that is ten times longer than the required control period. Or, for the sake of the argument, to run control loops over the network which are required to be reliable, when it is known that parts of that network are not reliable. But that object-oriented platform would be highly suitable as the architecture paradigm, if compounded with performance, dependability, and other non-functional requirement specifications, that would educate the system configurator as to how and where to run specific applications[Pow91].

In consequence, it is necessary to take non-functional requirements into account. The need for them is sometimes not obvious in early stages, but good sense implies that industrial system platforms should accommodate provisions for: fault-tolerance by replication, hard real-time domains, mobility and remote operation. This objective can only be fulfilled if taken into account very early in the system architecture conception.

3 RATIONALE FOR THE ARCHITECTURE

A realistic distributed industrial information system (DIIS) has to fulfill the requirements expressed in the last section: de-facto standards; SME needs; non-functional attributes. An effective DIIS must cover the whole information flow, from/to sensors and actuators to the managers. We address these issues in the discussion of our architecture.

The de-facto standard problem can be equated in the following way:

- Factory floor integration reasonably new, and reasonably well-served by CNMA framework: MMS, FTAM;
- management and "engineering" floor integration and applications long established; less well served by CNMA framework; prevailing UNIX "world" and UNIX networking (sockets,

TCP/IP, FTP, Yellow Pages, etc.), X11 Windows, NFS, OSF-DCE, etc.; a number of existing applications compatible with this world.

The SME problem can be equated as follows:

- Aim at an essentially centralised system at entry-level, which is *distributable*;
- Aim at a small-scale system at entry-level, which is scalable upon need.

Handling the above-mentioned non-functional attributes is a user requirement on one hand, since they are triggered by specific needs, and a technical requirement on the other hand, since one must ensure that the system does have the attributes. This latter part will not be addressed with detail in this paper. However, architecturally, we took the approach of defining placeholders, i.e. points in the architecture, where it would be possible:

- to act in order to achieve given attributes (e.g. replication for fault-tolerance);
- to break it in order to confine sections of the system where given attributes might be obtained (hard real-time domains).

In essence, the whole problem of materialising the abstraction of a CIME environment can be approached by devising specific domains, at two major levels: factory floor and management/engineering floor. The factory floor contains the front-end of automation applications; it requires safety and reliability, real-time performance and timeliness; responsivity and concurrency; autonomy and modularity (cells, etc.); in essence, it deals with *active* processes. The management/engineering floor contains the back-end of automation applications (stock-control, statistics, quality management and control, engineering, etc.); it requires security and availability; limited concurrency; information sharing; multi-user operation; virtual machinery; transactional support; in essence, it deals with *passive* data.

3.1 The information flow

In this section, we introduce the generic issue of the information flow and relevant repositories in the system, to reason about the several architecture options we may have available. Figure 1 gives an information flow view of the system. It is applicable to any generic automation system. Machine layers (sensor/actuator devices, computers) alternate with information layers (data), to illustrate the kind of information each device deals with, and how the information migrates between layers of abstraction. The figure does not locate yet where the information is physically. Shop floor devices handle information from or on behalf of the physical processes. As such, the transformation of this information into a computer tractable representation is a crucial point[KV93]. The abstraction chosen is in the scope of the MMS[ISO90a, ISO90b] manufacturing standard: the Virtual Manufacturing Device (VMD), which enables the encapsulation of any device in a set of data structures and control programs, with the corresponding ease of treatment, in a client-server way, by the system's higher level applications.

Let us start by observing the bottom-up trajectory. The VMD state represents what we call the *real-time information* of the system. This information aims at reflecting the state of the devices as accurately as possible. Around that raw data, specialised programs may build pre-processed data (composite variables, averages, etc.), necessary for the several automation applications. The management-level programs (production management, quality control, etc.) use the history of the process accumulated in a stable-storage database. They may also use the real-time information for state-less uses, such as to mimic the synoptics of low-level processes

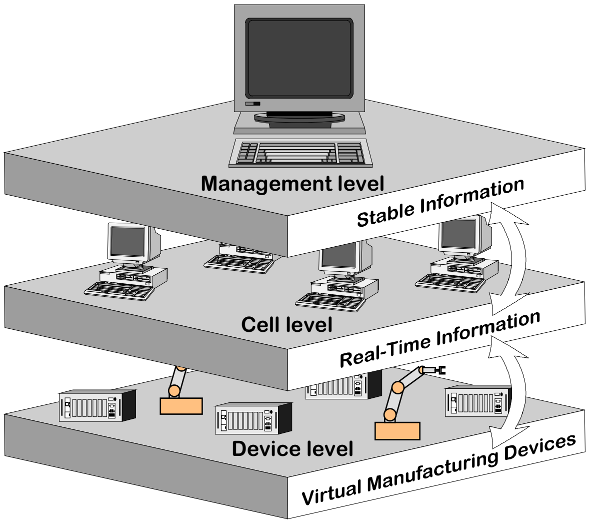

Figure 1: View of the information flow.

seen by the shop-floor controllers. The accumulated information is called *stable information*. Unlike the real-time information, which is re-written and re-calculated, this information is re-calculated and accumulated. Re-writing concerns update of the real-time VMD variables (a temperature, a valve state). Re-calculation concerns the update of complex variables belonging to the pre-processed data, like rates and variables deriving from expressions. Accumulation concerns gathering of statistics about successive states of variables, namely the afore-mentioned ones, or logs of the operation of machines.

The top-down trajectory concerns two essential roles the management level: command of the production or process; feedback upon information received from the process. The support required from the architecture is thus a command-based interface, materialised in this case by the client-server functionality supplied by MMS. Indeed, a VMD, besides structuring the state of a device, also offers an interface for the services the device provides. Besides, the notion is compoundable and hierarchical, that is, a cell controller can encapsulate (and control) a set of VMDs (for the devices that form the cell), and become itself a "cell-VMD" server on behalf of the latter, for the higher-level applications.

When considering the possible settings on which to implement such a system, several issues concerning heterogeneity and distribution arise. These impact how the structures shown in figure 1 are implemented, and where they are located.

3.2 Heterogeneity

Heterogeneity directly arises from the wide market offer of systems and equipments, in most of the cases with proprietary interfaces. The problem of heterogeneity can be viewed at different levels, such as the operating system level, the application level or the hardware level. In heterogeneous operating system environments the range of applicational software is limited by the availability of multi-platform software packages. At the application level, the problem of heterogeneity arises from the different and incompatible interfaces concerning data transfer offered by applications, which make sometimes impossible, for instance, the use of a specific database system. Finally, the existence of a wide range of different industrial devices may constitute an unsolvable heterogeneity problem for some SCADA systems.

The following is an enumeration of the possible problems concerned with heterogeneity in the system support of a DIIS:

- different operating systems;
- different implementations of the ISO stack, with different interfaces (access methods);
- different stacks (e.g. TCP/IP);
- existing or foreseen applications having in mind (the application or the system) interfaces other than ISO MMS and FTAM (e.g. Unix files, RPC, streams, sockets).

Operating system and support stack heterogeneity have been addressed by providing an uniform application programming environment, and/or porting of the application to the several target O.S.'s and stacks. Three facts deserve a special mention in the scope of industrial information systems, because they affect the way MMS and DIIS will become effective in industrial facilities.

The first is the fact that the machines (PLCs, PCs) foreseen to have MMS in the shop floor are still in limited offer. As such, measures for open MMS solutions and encapsulation of proprietary devices under a generic VMD would be welcome. We support the principle of grouping proprietary shop-floor device networks in clusters as appropriate, and encapsulating them as composite "vendor-X" VMD's.

The second are the interoperability and access method incompatibilities that have been found in different MMS implementations, despite more or less well succeeded interoperability demonstrations. Access methods to MMS by different vendors are so different sometimes that, strange as it may seem, an application running on MMS in one machine has to be greatly rewritten when ported to another MMS make. This problem is addressed by means of a harmonising communication platform, providing a uniform application programming interface. That is, the same format for MMS primitives is seen by applications running on any operating system and on any MMS make.

Finally, the third problem concerns the heavy bias toward the ISO protocols, of the MAP-CNMA stack. A solution for accommodating heterogeneity in communication stacks (for example, ISO and TCP/IP) would be welcome, for most distributed environments at the management level are UNIX (and thus TCP/IP) based nowadays. Our architecture has addressed the problem by porting MMS over TCP/IP. Related real-life projects have provided a set of ports for different systems and for both the TCP/IP and ISO stacks, as we discuss in section5.

Still, syntactic uniformity is not enough. Distributed programming cannot be supported on communications alone. The issue of distribution support has to do, in the context of a DIIS, with data consistency and availability, location and access transparency. These issues become complicated in real-life settings, with the need to accommodate existing applications. To be more specific, applications written for the UNIX world, on machines that do not have MMS, must be supported, because they are bound to constitute an important segment, both in the factory and in the management floors. The heterogeneity of DBMS must also be taken into account. The next section discusses our approach to achieve transparent and scalable distribution in heterogeneous environments.

3.3 Distribution

Information distribution is a major requirement in modern computational environments, in particular in industrial information systems. Unfortunately, many of the existing systems lack in providing efficient solutions for the problem of information distribution, since most of them are oriented towards centralised control and supervision at the cell level, and centralised databases at the management level. For the same reason, they are also incapable of solving the problems of scalability and expansibility.

Introducing distribution in an industrial environment poses a set of problems, that can be equated around three topics:

- inherent shop-floor machine distribution;
- real-time information distributiveness;
- stable information distributiveness.

A system such as depicted in figure 1, is based on two layers of information: the shop floor data, dynamic and real-time; the historical data, stable and cumulative.

The best way to get to an effective architecture, is to focus on achieving *distributiveness* of the information, vis-a-vis the evolution of the facility: expansion, layout changes, management changes, and so forth.

By distributiveness we mean the capability of the system support to become distributed in the measure of the industrial facilities needs. That is, a system may start by being centralised in a small facility, and then grow distributed without any basic change, in the measure that the facility is expanded. Moreover, several of these formerly centralised modules may grow to become a larger, distributed, infrastructure. By introducing the concept of distributiveness, we intend to fulfill one of the objectives of this architecture: make it suitable for SMEs.

3.4 Information distributiveness

It is perhaps evident at this point that our architecture was conceived in order to solve the distributiveness issue at major, well-identified points: the real-time repository; and the historical repository. Likewise, although not treated in this paper, this is the obvious place to address non-functional requirements as mentioned earlier: replicate given real-time repositories of sensitive cells, for reliability and availability; replicate historical repositories for the same reasons; set-up real-time repositories in a local hard-real time environment for effective distributed computer control, while still allow the relevant information to be included on the general operation of the DIIS; replicate historical repositories for performance reasons in access by distributed applications, and so forth.

Shop-floor data, resident in the VMDs, may be available to applications in either of two ways: (i) directly at the source; (ii) from an intermediate repository. Direct access (figure 2a) is made on demand, that is, applications dialoguing with the shop-floor real-time data (example, a data collection module), wherever they are, look for the information they need. It is also made by up-calls, via an event manager (for example, an alarm manager module). The applications do not share information at this level. For example, an event that needs to arrive at two client processes will be up-called to both of them. Information is dumped into the stable information repository (figure 2b), processed, and the results written again into stable storage, or output.

Alternatively, a state-based approach allowing some distribution and sharing at the real-time data level, may be followed, as depicted in figure 2b. It consists of creating a real-time intermediate repository. Applications handling process information address the latter, rather than the devices themselves. This provides for elegant shop-floor structuring, eases reconfiguration and expansion, and allows sharing by high-level applications of the same physical data, without overloading the controllers with repeated direct requests. On the other hand, the R/T repository is built by special modules with direct access to the shop-floor VMDs through MMS (figure 2b), either on request (periodically), or by up-calls, as will be discussed in section4. This approach may also accommodate in the same architecture time-critical control loops, provided that the repository of the relevant cell is a hard real-time one.

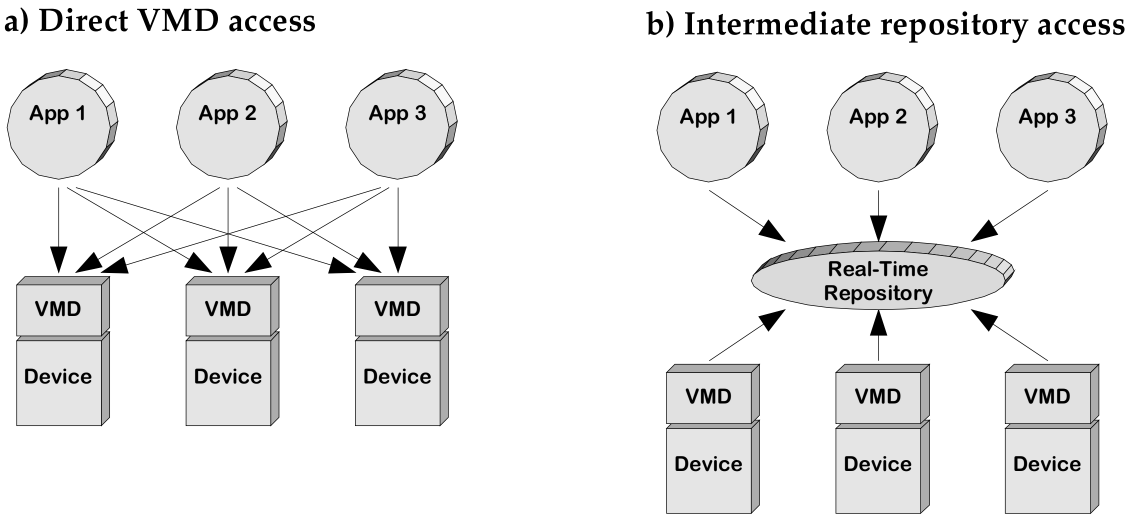

Figure 2: Shop-floor data access methods.

Stable information raises two relevant issues for discussion: storage and retrieval. Again under a user oriented perspective, three relevant policies for storing and distributing the stable information should be accommodated. They are depicted in figure 3. The idea is that if these three alternatives are contemplated, the migration from SME-sized systems to large-company ones within the same architectural framework will be safeguarded.

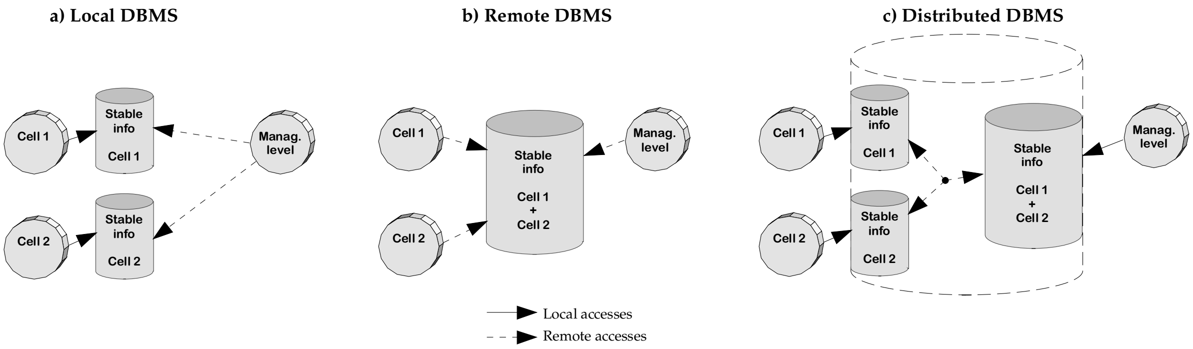

Figure 3: Stable information distributiveness.

Figure 3a depicts the isolated-repositories alternative. Historical data is stored in several local databases with no connection with one another. Each database and manager concern the context of a manufacturing cell, and they reside on the cell controller. This is a scenario of a small enterprise, where everything is integrated in a machine, or the (undesirable) situation of an SME that has grown meanwhile, without any integration. Figure 3b depicts the centralised-database alternative, used for several cells: the database manager serves a whole physical factory unit, holding information from several of the units concerned with the first alternative. The DBMS may reside in one of the cell controllers or in a database server. Each cell only accesses the information relevant to it, that is, the DBMS is a single physical repository, although it is seen as multiple logical repositories. This is the scenario of a medium enterprise, with a few cells, but limited informatic resources. The DBMS may become a bottleneck, as well as a single point of failure. Figure 3c depicts the distributed-database alternative. It can be seen as the scenario of figure 3a, migrated to a distributed DBMS, in order to interconnect all local DBs. What is gained with this? The cells retain their modular structure, and historical data is created and fed to the database on a local basis. However, the information is globally accessible. For the management level, this is an elegant method for any application, located anywhere, to access easily each and every cell history. For applications in general, an also elegant means to share stable information.

4 THE DISTRIBUTED PLATFORM

In this section we present the NavCim distributed platform, which aims at suppressing the major problems of industrial information system integration in heterogeneous and distributed environments, with scalability and expansibility requirements. We first describe the key concepts that guided the design of the proposed platform and then, an internal view of the platform components and its relationships is presented.

The already stated requirements of industrial information systems and the problems of most of the current available solutions when facing them, lead to the design of the NavCim platform. The NavCim platform offers functionalities for industrial process supervision and control, and support for the execution of management tasks, such as quality control, production reports and alarm supervision. To accomplish this, the NavCim platform presents an open profile, which is based in the use of standard solutions both for communication protocols and for information storage supports.

The heterogeneity problems at the operating system level are solved by using an operating system interface layer which provides software portability thus allowing the platform to be built on top of multiple operating systems. This layer is provided by the LSE (**L**ocal **S**upport **E**nvironment) package, described in [FRRV90].

The independency in terms of industrial device heterogeneity is achieved by using the industrial communications standard MAP/MMS [ISO90a, ISO90b], which uniformizes device access method and comprises advantages in terms of system distributiveness and expansibility. In the NavCim platform the MMS environment is provided by the SWCP (**S**ystem **W**ide **C**ommunications **P**latform) software package [RS92], which offers some operating system independent programming libraries as well as the entities required for remote communication over MMS.

With regard to information storage and access methods heterogeneity, the NavCim platform handles the problem, in general terms, by assuming that most applications have the capacity to access to the file system (which is a rather realistic assumption), thus enabling the use of files as a mean of data dissemination and guaranteeing, at the same time, compatibility among all applications.

Then, if the file system is a *distributed file system*, it becomes possible to grant file system access to any distributed application, thus enhancing data visibility. A file system such as the NFS (**N**etwork **F**ile **S**ystem) [Mic89] file system may be used. Since NFS is a standard and is included, by default, in most operating systems, its use becomes inexpensive and the overhead in terms of configuration and integration procedures is almost inexistent.

Real-time behavior of the NavCim platform is not a hard requirement. However, some real-time characteristics may be attributed to part of it, which may be important for some specific implementations. Also fault tolerance issues are not specifically focused in the basic platform architecture, but hooks were left to allow the platform to evolve in new directions, namely by incorporating replication techniques supported on group technology[KV93].

In the design of the NavCim platform, a hierarchical structure representing the industrial information system was conceptually defined. This hierarchy is composed of two basic levels — the device and the cell levels — plus a third level corresponding to the management level, which may or not exist in a real scenario. The first two levels are usually located in the shop-floor and the top level resides in the office-floor.

The device level is the bottom level of the structure. At this level one can find the devices or equipments that produce the supervision information, used in higher levels. Devices may be logically associated, forming production cells, as result of a determined productive process organisation. The device level is then formed by many of these cells, which may be supervised independently. Since MMS is used for communication between the device level and the upper level, the NavCim platform provides the infrastructure for this communication process. In fact, for communication purposes each device is seen as a VMD, which is accessed through MMS by clients in the NavCim platform. The VMD may be either built by the device itself (if it has the ability to communicate through MMS) or by a representative process (acting like a gateway between MMS and a specific device communication protocol).

The supervision of each factory cell is performed by a NavCim instance, called a NavCim **cell**. The communication between the NavCim cell and the devices (its VMD representation) is made through MMS, contributing to the desired distributiveness, scalability and heterogeneity support characteristics of the platform. Some of the advantages of using MMS are:

- Device characteristics independency;
- Device localisation independency;
- Automatic usability of MMS compatible devices;
- Any device may be connected.

The NavCim cell is responsible for data collection and is able, if needed, to execute some supervision and control tasks. Each cell builds a *real-time image* of all devices, and stores these images in *real-time repositories*, accessible by all applications [1]. These repositories consist of files that, by means of a distributed file system, may be visible in upper levels of the hierarchy or, if needed, in other cells at the same level. Just like raw device data, also pre-processed data, events and alarms are stored in real-time repositories and are accessible by all applications.

The NavCim platform defines a set of entities which are organised as separated logical modules and may be classified in the following three categories:

- **Support modules** — are independent software packages, mainly used for communication purposes.
- **Interface modules** — offer application programming support on top of the NavCim cell. These modules are not strictly necessary to build supervision applications, since it is always possible to access real-time repositories directly.
- **Internal** NavCim **modules** — constitute the NavCim cell core. They implement the main functionalities of the cell.

[1] This is consistent with a *state-based approach*, where real-time repositories contain the up-to-date state of manufacturing processes

The most important support modules, already mentioned in the text, are the SWCP, the NFS and the LSE. Two interface modules are defined: one for data access purposes and the other one for direct interaction with the cell, mainly for configuration purposes. Internal NavCim modules execute particular tasks needed for the success of the overall supervision and control process.

In figure 4 it is possible to observe how information flows inside each NavCim cell.

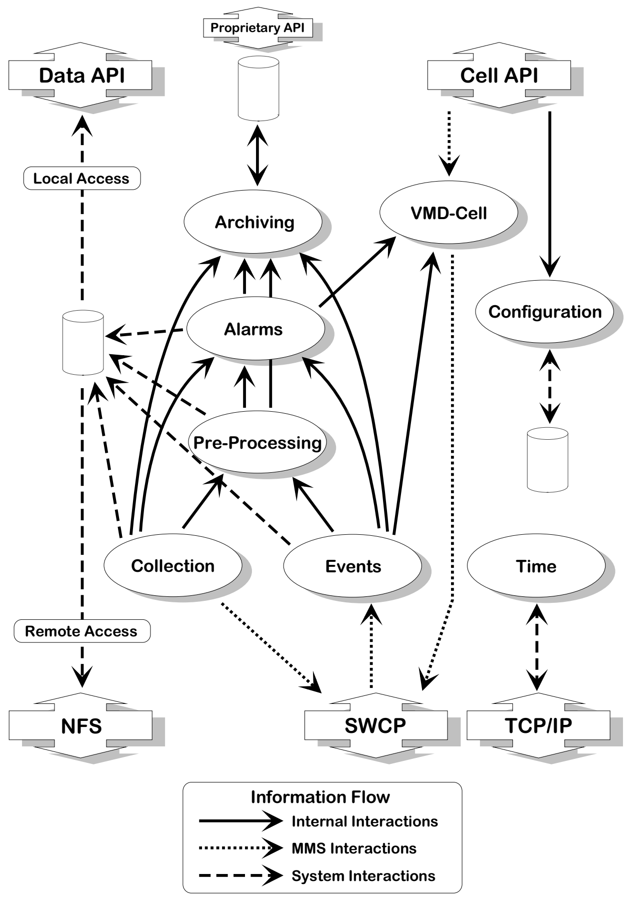

Figure 4: Information flow inside a NavCim cell.

Three basic types of interactions may be identified: internal (among platform modules), MMS (using SWCP module) and system interactions (using the TCP/IP stack or system calls). Following, a description of the attributes and functionalities of the depicted internal NavCim cell modules is dealt.

- **Configuration** and Initialisation — This module performs initialisation tasks, accordingly to configuration parameters read from configuration files. VMDs associations are established and other cell modules are executed. This module is also responsible for the execution of dynamic configuration modifications, requested via the cell interface module.

- **VMD-Cell** — Industrial device control can be accomplished through this module. It consists of a global VMD representation of all devices residing on a factory cell and is accessed through MMS by applications that need to manipulate process variables (usually for control purposes). Although it is possible to access device VMDs directly, all control information should flow through this module in order to preserve the benefits of an hierarchical architecture.

- Data **Collection** — This module is responsible for data collection and maintenance of real-time repositories freshness. Several collection activities may be executed in parallel, gathering one or multiple data values which are stored in a single real-time repository. If configured so, data may also be forwarded to the pre-processing module.

- Data **Pre-processing** — This module offers a set of data processing functions that may be used by programs to generate real-time information about the manufacturing process evolution. Pre-processed data may also be used as input for the alarm management module. This module may be used to generate statistical data for Statistical Process Control.

- Data **Archiving** — Historical data is generated by this module, which stores real-time (raw or pre-processed) information in a stable repository.

- **Event** Management — This module is responsible for receiving and managing MMS events sent by cell VMDs. Events are disseminated through real-time repositories (following a state-based approach, just like collected and pre-processed data), but may also be forwarded to applications via MMS, following an event-based approach. Event pre-processing may also be carried out by the pre-processing module.

- **Alarm** Management — It is possible to program alarm conditions in the NavCim cell, which will be handled by this module. Raw, pre-processed or event data may be used to generate alarm conditions. Upon alarm triggering several actions may be taken, including generation of alarm reports (either via real-time repositories or via MMS events) and device control actions (manipulating VMD variables via the VMD-Cell module).

- Global **Time** Service: — This module implements a synchronisation protocol among all NavCim cells. The time service is used by other modules as the basis for timestamp generation.

5 EXPERIENCE GAINED WITH DINAS-DQS

The objective of DINAS-DQS[2] was the application of CNMA based communication at an industrial environment to support a Distributed Quality Control System. The project demonstrated firstly the advantages from developing CIM applications in SME's using CNMA communications and secondly derived a set of guidelines for using CNMA in SME's.

The manufacturing environment of the industrial partner, on which the infrastructure was built, consisted of two manufacturing sites, linked by a MAN (FDDI backbone). Shop-floor operations in each of the sites were performed by a heterogeneous assembly devices as well as testing devices that were linked by a LAN (Ethernet) to support soft real-time operations and information gathering for quality control and management.

During this project, INESC was mainly involved in the definition of the user requirements on networking infrastructure, in the specification of the internetworking architecture and services, in the network infrastructure profile, and in the network and services implementation.

In figure 5 we present a general view of the pilot architecture.

[2] Esprit Project 6779. Partners: INTRACOM (GR), INESC (P), IPK (D), INTRASOFT (GR), HYPERION (IRL).

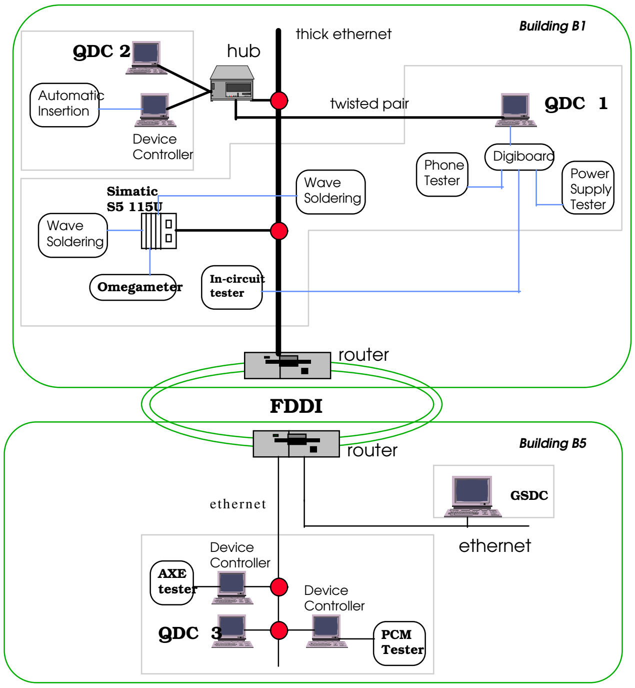

Figure 5: DINAS-DQS Pilot Architecture

The quality application was distributed between the General Supervisor Decision Center (GS-DC) and the Quality Decision Center (QDC). The GSDC comprised the Data and Alarm Definition & Maintenance, the Alarm Exploitation, the Collected Data Processing and the Statistical Processing of Data. The QDC comprised the Collect Data Application, the Statistical Processing of Data, the On-line Alarms and sporadic VMDs implementations. Data was collected by the QDC from the different VMDs and was stored in the QDC database for further exploitation.

In the shop-floor, the communication services were based on the client/server model in order to allow message, file or event passing between applications, and database record access. These services were built around the SWCP, offering an homogeneous API syntax, throughout the various target systems. The manufacturing cells were conceptually represented as a set of VMDs, which were accessed via the MMS protocol.

All the relevant shop-floor data collected from the VMDs, were stored in an Ingres database (directly or post-processed). The information flow from the QDC to the GSDC was mainly performed through the proprietary Ingres DBMS, with its communication support mechanism which was based on TCP/IP.

As discussed earlier on, support for heterogeneous communication stacks and operating systems is an important criterion of effectiveness of industrial information systems. In figure 6 we present all communication profiles considered.

The network infrastructure was structured along domains, that confined different functional and operational requirements, with respect to size (distance and number of nodes), expected load, as well as service availability, security and timeliness, i.e. dependability and real-time constraints.

Thin Ethernet LAN segments were selected for equipment interconnection inside each domain. This choice was due to the small number of nodes per domain, limited physical span, low expected load, moderate timeliness requirements and also justified by financial and commercial reasons.The separation of manufacturing domains in individual LAN segments presented real advantages with respect to reliability and isolation of failed segments.

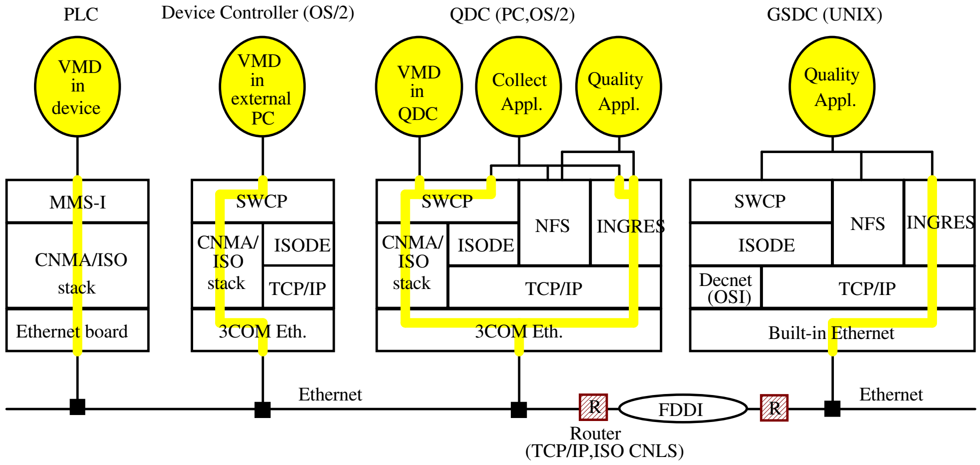

Figure 6: DINAS-DQS Communication Profiles

The geographical span of individual domains along the INTRACOM's premises, led to the need of their interconnection. For expansion and containment reasons, this interconnection was performed through network routers linked by means of a FDDI backbone. The advantages of this option were: high throughput and speed; low latency; high reliability; the potential capability of performing backbone interconnection between all the buildings in INTRACOM.

5.1 *De-facto* Standards

In the scope of DINAS, INESC was particularly involved in the pragmatic view of integrating the CNMA services with the *de-facto* standards from the Unix communication world: NFS, TCP/IP, etc.

The NFS-based approach offered to the applications a simple and transparent file distribution mechanism which could support the bottom-up information flow. This mechanism, despite its restrictions in very demanding environments (high load, throughput, etc), was perfectly indicated (as proved in one of the DINAS testbeds) for INTRACOM distributed quality system.

During the project, INESC developed a SWCP agent over TCP/IP for Unix and OS/2 operating systems. This agent uses the ISODE package to implement ISO-OSI layers 4 to 6 over TCP/IP. Although the SWCP agent is not a full ISO stack— hence it cannot be used to communicate directly with devices that implement such a stack— it has some important advantages which justified the development:

- The use of TCP/IP which is a well known protocol, tested and less expensive than the correspondent OSI protocols;
- The use of ISODE (which is free software);
- The actual availability of TCP/IP in almost any operating system, and the way ISODE is made to allow easy ports to different environments, makes the SWCP agent easy to port to almost any operating system (including some PLCs).

6 Conclusions

In conclusion, we have discussed the design of a *distributed industrial information system platform* (DIIS). Three user-oriented requirements have guided our design options: de-facto standards;

the SME problem; non-functional attributes, amongst which scalability and modularity to accommodate special needs (fault-tolerance, real-time, mobility, etc.).

We proposed an architecture which integrates the whole information flow of industrial facilities: bottom-up (supervision) and top-down (command/control). It spans from the sensor/actuator device levels to the high-level management applications. In essence, our platform is an integration framework, and because of that, treats heterogeneity, distribution, and scalability as its key issues.

During the design of NavCim, we had the opportunity of experimenting these ideas, by integrating an early version of the platform in a real-life setting, a distributed quality management system.

As future work, we plan to address the issue of mobility, by inserting GPS-based guidance in mobile cells, and GSM-based communications, to support mobile industrial and mission-critical information systems. In addition, and in relation to the increased need for reliability, we will develop fault-tolerance extensions, by allowing selective replication of information both at the cell and database levels.

References

[Bee89] D. Beekmann. CIM-OSA: computer integrated manufacturing - open system architecture. *International Journal Computed Integrated Manufacturing*, 1989.

[BGG+91] Allan Bricker, Michel Gien, Marc Guillemont, Doug Orr, and Marc Rozier. Architectural issues in microkernel-based operating systems: the chorus experience. *IEEE computer communications*, July 1991.

[But86] Alan Buttle. Practical open systems interconnection over ethernet. In *NETWORKS'85*, pages 113–116, 1986.

[CNM93] CNMA. CNMA Implementation Guide, Revision 6.01. Technical report, ESPRIT Project 7096, July 1993.

[Coh87] Marc Cohn. The applicability of the emerging ans fiber distributed data interface (FDDI) for a distributed avionics architecture. Technical report, X3T9.5/87, working document, February 1987.

[Com88] Douglas E. Comer. *Internetworking With TCP/IP: Principles, Protocols, Architecture*. Prentice Hall, Stevenage, 1988.

[Com91] Douglas E. Comer. *Internetworking With TCP/IP: Principles, Protocols, Architecture*. Prentice Hall, 1991.

[FFH+90] Jean Fenart, Marc Fievet, Christian Huitema, Bernard Martin, Annie Remille, and Guy Vaysseix. OSI and TCP/IP protocols on a Unix system V. Technical report, GIPSI-SM, 1990.

[Fou90] Open Software Foundation. Distributed Computing Environment. overview. Technical report, OSF, The Open Software Foundation, 11 Cambridge Center, Cambridge,MA02142, 1990.

[FRRV90] H. Fonseca, L. Rodrigues, J. Rufino, and Paulo Veríssimo. Local Support Environment: User Specification. Technical Report RT/50-90, INESC, Lisboa, Portugal, August 1990.

[GAT92] GATIE. Development of cim in portugal. Technical report, Ministério da Indústria, 1992.

[Int88] *The BITBUS Interconnect Serial Control Bus*, July 1988.

[ISA85] Instrument Society of America. *ANSI/ISA S72.01-1985, Proway-LAN Industrial Data Highway*, 1985.

[ISO90a] International Organization for Standardization. *MMS Specification - Part 1: Service definition*, 1990.

[ISO90b] International Organization for Standardization. *MMS Specification - Part 2: Protocol specification*, 1990.

[JN90] E. Douglas Jensen and J. Duane Northcutt. Alpha: A non-proprietary os for large, complex, distributed real-time systems. In *Proceedings of the IEEE Workshop on Experimental Distributed Systems*, pages 35–41, Huntsville, Alabama, October 1990. IEEE.

[JW86] Dittmar Janetzky and Kym S. Watson. Token bus performance in MAP and Proway. In *Proceedings of the IFAC Workshop on Distributed Computer Protocol System*, 1986.

[KBS91] Don Krieger, Gerald Burk, and Robert Sclabassi. Neuronet. *IEEE Computer*, March 1991.

[KDK+89] Hermann Kopetz, Andreas Damm, Christian Koza, Marco Mulazzani, Wolfgang Schwabl, Christoph Senft, and Ralph Zainlinger. Distributed Fault-Tolerant Real-Time Systems: The Mars Approach. *IEEE Micro*, pages 25–41, February 1989.

[KV93] Hermann Kopetz and Paulo Veríssimo. Real-time and Dependability Concepts. In S.J. Mullender, editor, *Distributed Systems, 2nd Edition*, ACM-Press, chapter 16, pages 411–446. Addison-Wesley, 1993.

[MAP85] *Manufacturing Automation Protocol Specification V2.1*, March 1985.

[McMS92] J. M. Mendonça, S. McCarthy, and J. W. Schulte. Integration of Shop Floor Control Applications in a Real-Time Data Collection Environment. In *Proceedings of the Eighth CIM-Europe Annual Conference*, Birmingham, U.K., 1992.

[Mic89] Sun Microsystems. NFS: Network File System Protocol Specification. Technical Report RFC 1094, Sun Microsystems, Mountain View, CA., March 1989.

[ODP87] ISO/TC97/SC21 N1889. *Proposed Revised Text for the New Work Item on the Basic Reference Model for Open Distributed Processing*, 1987.

[PCB+91] David Powell, Marc Chereque, Peter Barret, Douglas Seaton, Gottfried Bonn, and Paulo Veríssimo. The delta-4 distributed fault-tolerant architecture. Technical report, LAAS-CNRS/Esprit Delta-4, 1991.

[PFB87] *Profibus Proposal to ISA SP50*, December 1987.

[PJ85] J. Postel and Reynolds J. File Transfer Protocol (FTP). Technical Report RFC 959, October 1985.

[Pow91] D. Powell, editor. *Delta-4 - A Generic Architecture for Dependable Distributed Computing*. ESPRIT Research Reports. Springer Verlag, November 1991.

[RS92] C. Rang and W. Schönewolf. Communication Platform Prototype — Information. Technical report, IPK, Berlin, 1992.

[Ver93] Paulo Veríssimo. Real-time Communication. In S.J. Mullender, editor, *Distributed Systems, 2nd Edition*, ACM-Press, chapter 17, pages 447–490. Addison-Wesley, 1993.

[Zim80] Hubert Zimmermann. OSI Reference model - The ISO Model of Architectur for Open Systems Interconnection. *IEEE Transactions on Communications*, COM-28(4):425–432, April 1980.