DEVELOPING INTEGRATED PERFORMANCE MEASUREMENT SYSTEM - USING MDA APPROACH

Souhail Sekkat[(a), (b), (c)] , Jean Luc Paris[(a)], Khalid Kouiss[(a)], Janah Saadi[(c)] , Laurant. Deshayes[(d)]

[(a)] LIMOS – IFMA Campus de Clermont-Ferrand/Les Cézeaux - BP 265 - 63175 Aubière Cedex, France.
[(b)] ERCSSP- ENSAM Marjane II Beni M'hammed, BP 4024, Meknes Alismailia, Meknes, Maroc.
[(c)] LAP - ENSEM Casablanca B.P. 8118, OASIS. Route d' EL JADIDA, Casablanca.
[(d)] MANBET TECHNOLOGY : 5 Avenue Abou Taeib Al Moutanabi Residence Walili Apt 13, Fes.

[(a)]paris@ifma.fr, kouiss@ifma.frail, [(b)]s_souhail@hotmail.com [(c)]janah1@menara.ma [(d)]laurent.deshayes@gmail.com

ABSTRACT
In an industrial context defined by more acute competition, performance measurement becomes a control tool. The Manufacturing Execution System (MES) software achieves control and execution of production functions, as Performances analysis. However, to choose the adequate performance indicators and to implement them is a difficult problem. Indeed, the enterprises need methods to specify and to install their Performance Measurement System (PMS). In this paper, we propose a methodology of performance indicators implementation. We use the Component Oriented Programming with a Model-Driven Architecture (MDA), to generate applications' final code. This method facilitates the design of Performances Measurement System and its implementation.

Keywords: Manufacturing Execution Systems MES, Performances Measurement Systems PMS, Component Oriented Programming COP, Model-Driven Architecture MDA.

1. INTRODUCTION

Nowadays, the level of integration determines the effectiveness of modern technical systems. Application of the system engineering results in that systems become distributed, and consist of a number of physically or logically distributed components that constitute the system to reach the common goal. Communication between such components is a crucial issue.

The performances analysis is among the important components of computer control system. The development and the implementation of a Performances Measurement System (PMS) is a difficult problem. Indeed, to design and implement a Performances Measurement System (PMS), we must properly model identities, attributes and operations of individual objects as well as the sequence of operations and data flow in a shop floor control system. What will allow, thereafter, the time follow-up of the key performance indicator (KPI).

Therefore, in this paper, we present a shop floor control model and its implementation using Object and Component Oriented Approach.

Several researchers have applied Object-oriented and Component-oriented concepts in implementing manufacturing execution systems MES. For instance, (Yang and al 2000) applied object-oriented programming (OOP) to develop a CIM system. In addition, a distributed automation and control system has been implemented in (Krassi 2002). For system integration a middleware CORBA were used. Another work was addressed on applying COP methodology to develop a controller, in robotic palletization area (Chiron 2009).

Although various works related to the modeling and implementation of shop floor controllers were presented, the implementation of PMS was not addressed by those researchers. MDA approach can be used to support this issue. Basically, it focuses on:

1. Identifying objects and their related properties,
2. Building the relationships among objects,
3. Describing the data transformation within the system.

The main body of this article is organised as follows. In the next section, the methods of PMS software design are presented and discussed. Then, proven industrial technology used in the process of application tool integration is presented. After then, proposed approach for specification and implementation of PMS is outlined. The article closes with a practical example and drawing a conclusion.

2. THE DESIGN OF PERFORMANCE MEASUREMENT SYSTEM (PMS)

The rapid progress of computer technologies brings shop floor control functions into a new area. Many advanced functions have been introduced as the results of improving computing power. These functions include

real time scheduling, networking, cell coordination and Performance analysis (Halang and al 2006).

In this section, we will present the PMS software and the different methods used for its design. First, we are going to describe the Computer Integrated Manufacturing (CIM) pyramid and to position the PMS in this pyramid. Then we will define the Overall Equipment Effectiveness (OEE). It is the PMS we will implement. Finally we are going to describe the methods used to develop shop floor control system.

2.1. CIM Pyramid

Since the functions of shop floor control have become more complex than before, several control architecture and models were proposed. One of the most famous models is NIST AMRF (National Institute of Standard and Technology / Advanced Manufacturing Research Facility) Proposed by National Bureau of Standards.

The AMRF model is applying hierarchical control architecture; it is composed of five-level hierarchy: facility, shop, cell, workstation and Actuator/Sensor. Each level has its specific functions.

The business functions such as; cost accounting, aggregate planning are executed in the facility level. It is done generally by ERP software. Shop level is responsible for coordinating production tasks including resource allocation and task assignment. The tool used at this level is The Manufacturing Execution Systems MES. Cell level is subject to sequence similar parts in batch jobs and supervises supporting tasks such as material handling and transportation. The supervision is achieved, generally by SCADA software A workstation level is responsible for monitoring the execution of production tasks. It consists of a robot, a fabrication station or a material storage buffer.

The Manufacturing Execution Systems MES provide a module for performances Analysis. Within the functions of this module, there is computation of performance indicators and display of scorecards functions (MESA, 2000). The PMS is based on the computer controller systems for:

1. The data collection (starting time, break times, completion time, products quality…),
2. The computation of performance indicators and
3. The display of scorecards.

2.2. OEE Overall equipment effectiveness

The more utilized PMS, is the OEE Overall equipment effectiveness measures, there are three underlying metrics that provide understanding as to why and where the OEE gaps exist.

These measurements are described below:

- Availability: The portion of the OEE Metric represents the percentage of scheduled time that the operation is available to operate. Often referred to as Uptime.
- Performance: The portion of the OEE Metric represents the speed at which the Work Center runs as a percentage of its designed speed.
- Quality: The portion of the OEE Metric represents the Good Units produced as a percentage of the Total Units Started. Commonly referred to as First Pass Yield.

OEE is computed as follows:

$$OEE = Availability \times Performance \times Quality \quad (1)$$

To implement the PMS, we must integrate it within the control system. Indeed, it should collect the data, from the shop floor control and send the performance indicators to the business information system.

2.3. The design of Integrated Production System

A design of an integrated production system is a big project that includes several jobs; programmers, automation engineer, mechanics… A design of shop floor controller is achieved through several steps:

- Definition of needs
- Product and process Analysis
- System Specification
- Equipments installation
- Controller Implementation
- Operation and maintenance

The main process steps are the specification and the implementation.

During the Specification, it is necessary to describe functionalities of equipments (production system) and the controller (information system). Thus, modeling methods are used. Business process modeling is the activity of representing a company processus, so that the current process may be analyzed and improved. BPM is typically performed by managers who are seeking to improve process efficiency and quality. It allows the construction of models of a determined part of an enterprise.

The evolution of the complexity of information systems, the worry of optimization of software applications design drove the scientists to develop some modeling methods. The classic modeling methods are ones that describe the Information system and proceed to separate treatment of data and processes. The object-oriented methods are new approaches of development. It considers a system like a set of objects in interactions (every object achieves or undergoes some operations), examples of these methods (UML, OMT) (Kettani and al 1999). The methods dedicated to distribute systems are those allowing the development of multi agents systems (with distributed execution objects).

Implementation: It is a setting step, which consists in the choice of the machines, control software's, the programming of supervisory control interfaces and tests.

Shop floor control must be integrated system. Interoperability requirement is difficult to achieve. Indeed, many companies selling industrial software tools and hardware, such as distributed control systems, SCADA (Supervisory Control And Data Acquisition) packages, programmable controllers, etc., provided external interfaces for interoperability but employed proprietary protocols requiring custom software development. Integration of applications meant substantial development effort to provide adapter layers so that applications could operate collaboratively and share data.

3. APPLICATION TOOL INTEGRATION

To harmonize different vendors' solutions (non-proprietary) for distinct information domains accomplishing the engineering requirements, it is common sense that the enterprise automation system might rely upon a common architecture to enable open standards interoperability.

The evolution of the Internet has played a major role in the area of application interoperability by providing an infrastructure that links applications, businesses, and users. It has promoted the adoption of standards, such as XML for data representation, and the use of common software protocols, such as TCP/IP and SOAP (Simple Object Access Protocol). These advances combined with the adoption of industry standards, such as OPC, by major equipment and system vendors has given end-users the means to greatly simplify the sharing of data at all levels of an enterprise, from the production level up to the executive level. Indeed, international non-profit organizations, like OPC Foundation (OPC – OLE for Process Control) and OMG (Object Management Group) cooperate among themselves to enable the development of open and consistent automation system architectures. Examples of such efforts are OPC UA (Unified Architecture) and CORBA (Common Object Request Broker Architecture).

In this section we will present the two more used standards for interoperability of industrial systems: the OPC standard and the CORBA bus.

3.1. The OPC standard

The rising of OPC technology made the HMI software manufacturers have to develop only a driver for communication with devices, differently of the way before, which each manufacturer had to develop proprietary drivers for supporting its devices. In industrial automation, companies like Rockwell Automation, Simens … wrote their own HMI software and a proprietary driver for each industrial device including every PLC (Programmable Logic Controller) brand in order to retrieve process data. The standardization of these drivers was developed by a group of manufactures that have improved the OPC model due to them additions based on past experiences. It is mentioned for example the navigation system of the OPC tags through tree structure and its division by sectors.

The OPC standards are based on *Microsoft* Object Linking and Embedding (OLE), Component Object Model (COM) and Distributed Component Object Model (/DCOM) technologies and provide the foundation for developing OPC compliant software. In order to address the specific needs of the process industry, the original OPC standards for process data acquisition has been extended through the following sets of specifications: OPC-DA (data access for real-time data), OPC-HDA (historical data access), OPC-AE (alarms and events), OPCXML, and OPC Batch (OPC Task Force 1998).

Since 2003, OPC Foundation and a group of IHM and device manufactures have been working in a joint venture to define and implement a new OPC Specification. This new Specification is the OPC UA. The purpose of OPC UA is to provide enhancements for existing and next generation OPC products in the areas of security, reliability, and interoperability. OPC UA is designed to unify existing OPC specifications into an environment that will leverage Web-based technologies and standards such as Web Services, WSDL (Web Services Description Language), XML and SOAP (Simple Object Access Protocol).

The OPC UA is based on SOA (Service Oriented Architecture) through Web Service that transports XML data, which provides the communication among different software of different platforms, providing interoperability. XML lets interoperability of different software of different platforms to transfer information using a common language. Since XML is a meta-language, it is possible to create new languages to make a standard talk. Thus, OPC UA is the key to improve the transformation of simple data in knowledge, due to the open high-level mechanism, open the doors to store the information by an easier way for MES (Manufacturing Execution System) and ERP (Enterprise Resource Planning) applications.

3.2. CORBA-BASED AUTOMATION SYSTEMS

The Common Object Request Broker Architecture (CORBA) is a widely recognized by the control community object-oriented open-standard-based middleware for distributed systems integration. CORBA supports interoperability, portability and reusability of systems' components. There is a number of CORBA implementations and mappings to various programming languages that are available from different vendors. Also, its real-time and fault-tolerant specifications have been developed recently (Curtis, Stone and Bradley 2009).

A CORBA system is based on client/server and oriented object model. The different components of this model are:

- **The client application** it is a program that invokes the methods of the objects through the CORBA bus.

- **The object interface** (Stub) allows clients to use objects without compile-time knowledge of their interfaces. It is generated by the IDL compiler to provide a "proxy" to an object to make it appear local to the client.
- **ORB (Object Request Broker)** is the communications system which passes requests to objects, and results back to clients.
- **Object Adapter**: provides services to implementations (e.g. registration, management of object's lifetime).
- **The object implantation** is generated by the IDL compiler to invoke the implementation using a set of conventions. It is special ORB functionality which allows invocation of implementations without compile-time knowledge of their interfaces.
- Implementation: code to implement the object.

These abstract notions result in technological components provided by the CORBA norm.

CORBA component's interface is described in a neutral Interface Definition Language (IDL). CORBA IDL is a declarative non-programming language; it does not provide any implementation details. The methods that are specified in IDL can be implemented with any programming language such as C++ and Java using rules defined by OMG. The Object Request Broker (ORB) is an object bus that provides a transparent communication, means for objects to send message to and receive message from other objects.

The General Inter-ORB Protocol (GIOP) provides interoperability (a set of message formats) between ORB implementations from different vendors. The GIOP can be mapped onto different telecommunication protocols for transport protocol independence. Thus, the Internet Inter-ORB Protocol (IIOP) is a mapping of GIOP onto the TCP/IP protocol stack.

From the automation systems viewpoint, CORBA is a layer for abstracting communications among components of the system, and a means for systems integration. Various components, which have different functionality, hardware and software, but compatible CORBA-interface, are plugged into the ORB but for higher level of interoperability and integration.

Finally we conclude that the requirement for data to be available across all levels within an enterprise is increasing. The business information technology markets have been addressing the challenges of interoperability and application integration for many years through Enterprise Application Integration (EAI). The trend is towards distributed architectures with increased interoperability among applications from multiple vendors.

4. THE PROPOSED APPROACH

The proposed approach can be used to develop a manufacturing execution system (MES). It is an iterative approach; the technical details are differed until the implementation step.

As a first step, we propose to model the system by using UML language (Kettani and al 1999). We adopt the use cases diagram to elaborate the functional model of the system. The use cases identified are the different activities proposed by the ISA S95 standard (ISA 2000):

- The product definitions and production resources management
- The detailed production scheduling, production dispatching and execution.
- The production data collection, production tracking and production performance analysis.

As far as we are considered, we deal with big interest with production data collection and performances analysis activities.

The products, processes, workforce, and other elements in manufacturing are modeled, in UML as "objects" in computer based graphical diagrams. In the static model, we use class diagram, which describes, the major structure of objects in a system including the identities, attributes and operations of individual object.

The staffs that exploit the production cell (operators) as well as the facilities (machines) are modeled by a class RESSOURCE "figure 1". The factory is subdivided in REGION that regroups the facilities treating a products family. The raw materials, Work in Progress and finished products are described by a class MATIÈRE. The process is described by a class OPERATION. The methods used to execute these operations are described by the class GAMME. The performance information of the different operations is stocked in the data base RAPPORT.

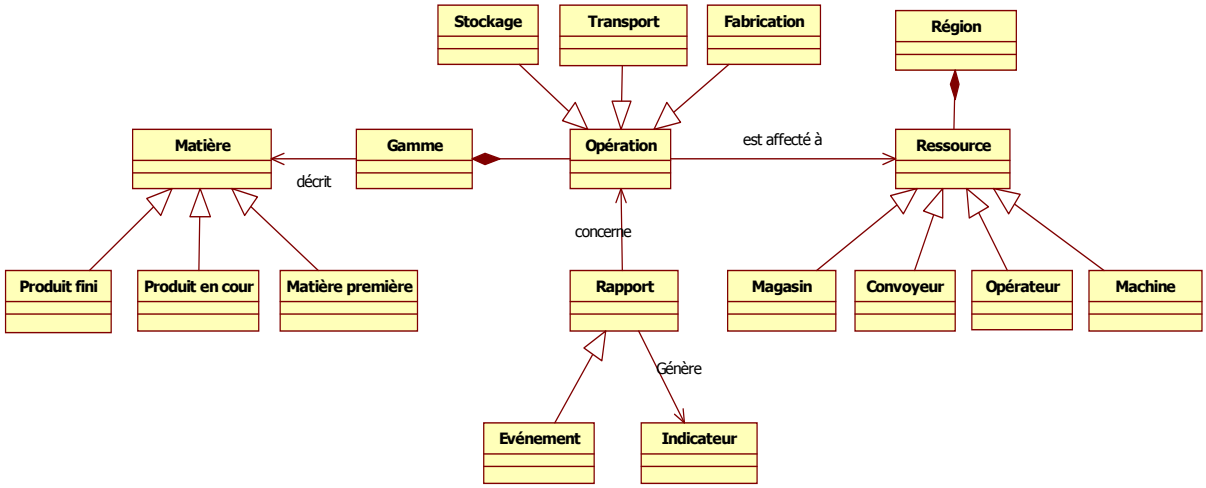

Figure 1: Class Diagram of the ENSAM Cell PMS.

The dynamic model describes the functions of the objects and their interfaces that they share with the other objects. Thus, we propose the use of sequence diagrams and state machine diagrams, which show the sequence of events both from a product perspective and from an information transaction viewpoint.

The script achieved by the SIP module of an MES software is as follow:

1. The class **Operation s**ends data collection demand to the class **Resource**.
2. The class **Resource** collects the data and sends back them in the **RAPPORT** class, the collected data are the starting time and the completion time of each operation as well as the mistake messages of the machine.
3. If there is an abnormal situation, the collected data will be sent back to the class **Operation**.
4. To analyze the performances, the **RAPPORT** class will collect the following shop floor data. The collected data will permit to analyze the performances and to compute the performance indicators.
5. The report of production will be produced and will be sent to management system.

As a second step, we propose to adopt a Component-oriented approach and use the MDA for the mapping of the UML models to the information system. Indeed, the proposed UML models are platform independent (PIM). It is necessary, therefore, to select a middleware platform that allow an interoperability of the distributed system, and refine the engineering models of the system in technical models, on describing how the platform is used to accomplish the interoperability requirements. In the context of MDA, the technical models are also called Platform Specific Models (PSM).

We propose to use Enterprise Application Integration (EAI) architecture, to allow interoperability, application integration and PMS implementation. (EAI) is an integration framework composed of a collection of technologies and services which form a middleware to enable integration of systems and applications across the enterprise. The middleware connects to applications through a set of adapters, also referred to as connectors. These are programs that know how to interact with an underlying business application, using different communication protocols. It acts as a workflow between the applications. We propose an integrating infrastructure based on computer standards, such as XML (data representation technologies) that allows the different heterogeneous applications to communicate by sending and receiving messages.

The Integration Platform, is therefore centered on a message server, it regroups the components that support the interoperability of tools. It essentially provides both data centralization and communication services that allow tools to share data and communicate between themselves. In addition to communication services, the integration platform contains a user interface module for indicators display. It hides all aspects of transport protocols from the component that it interfaces.

As an implementation step, we propose to elaborate, deployment diagram and component diagram. Deployment diagram serves to model the hardware used in system implementations, and the execution environments and artifacts deployed on the hardware. Component diagram depicts how a software system is split up into components and shows the dependencies among these components.

We must identify system components and affect them with responsibilities. For this, we will firstly make an external view representation, named black box view, and we will then, retail the component internal behavior by white box representation (Kouiss and al 2006).

The responsibilities of the KPI component are; data collection (machines status, product quality, execution time …), data verification, reaction when anomalies occur. In addition, KPI such as equipment utilization rate will be computed and displayed in business information system window.

5. PRACTICAL EXAMPLE

The proposed method is applied to the flexible machining cell in ENSAM. We will develop a PMS for its control. The system is organized around a belt trolley which allows a routing of the part toward a CNC milling machine "figure 2". A jointed-arm robot is used to load and unload parts for CNC machine.

- A 3 axes milling machine CNC MILL 55 with FANUC 21 controller is used to produce parts.
- A 5 axes industrial Robot RV-2AJ with an air gripper is used to load and unload of the CNC machine. The robot controller is a RISC processor, 64 bits, multitask and real time operating system. Robots programming is done by (Cosirop) software.
- A buffer with belt trolley, sensors and air stoppers, takes charge of storing and routing raw materials and finished parts. The belt allows a routing of the part. It is moved by electric motor. The storage system is controlled by the robot controller.

Figure 2: the ENSAM machining Cell.

In the control architecture of the system, a SCADA software (COSIMIR Contrôle) is installed in the top-level computer. An Ethernet network is connecting several machine controllers. The CNC machine is equipped with a DNC link based on RS-232 and the robot with Ethernet extension "figure 3".

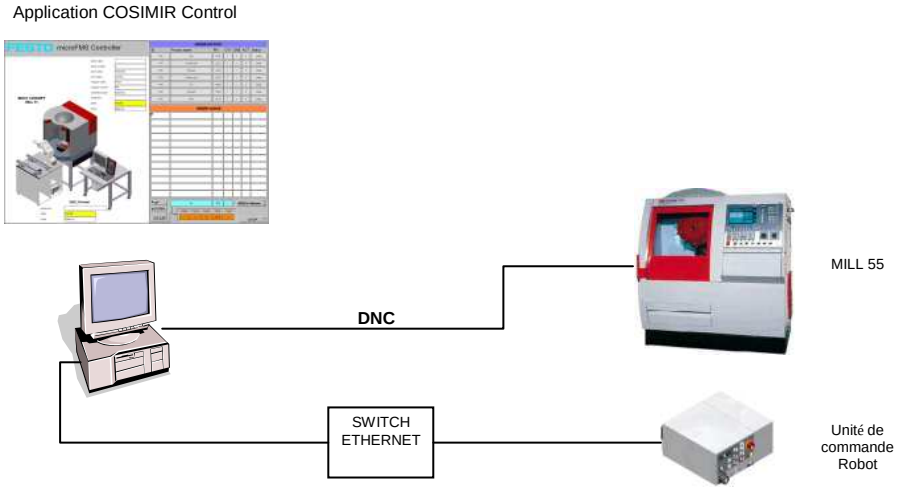

Figure 3 : Cellule flexible (partie commande).

The PMS must communicate with the SCADA software COSIMIR, it must also communicate with a MRP software.

Let us consider a work order sent by the MRP Software to the cell controller COSIMIR. The cell controller will select the production routine of the considered part from the Routine File database. It will check about the resources status. If all necessary resources are available, it will determine adequate material and start the tasks.

The strategy of integration was to provide a common software infrastructure (Integration Platform) that would allow a wide variety of application tools to communicate and share data, while minimizing the specific development for each tool. The heart of the Integration platform is SCADA software **COSIMIR Control**. It is a real-time, object-oriented development environment.

It is necessary to ask the COSIMIR software to collect the status machine during each process step. Thus, we elaborate an UML component diagram. We intend to implement the components in Visual C++. The data files required by the system like work orders, the routing, the workstation files, are stored in Access database.

Each UML class diagram proposed at the modelling step will be transformed in C++ code during the implementation step. Each class attribute will be declared like a variable. In addition, each class operation will be described like a method that executes the considered task.

6. CONCLUSION

The use of Object and Component approach, proven industrial technology coupled with message oriented middleware considerably eased the process of application tool integration. it allowed a better flexibility of the PMS.

Although the proposed model is focused on a CIM system in the educational environment, several concepts developed from this model still can be applied in the shop floor controller utilized in other environment.

Applying the techniques of business application interoperability to process applications implies that many other issues must be addressed when considering interoperability especially those related to safety and performance.

GLOSSARY

CIM Computer Integrated Manufacturing
COP Component Oriented Programming
CORBA Common Object Request Broker Architecture
EAI Enterprise Application Integration
ERP Enterprise Resource Planning
HMI Human Machine Interface
KPI Key Performance Indicator
MDA Model-Driven Architecture
MES Manufacturing Execution System
OEE Overall equipment effectiveness
OOP Object Oriented Programming
OPC OLE for Process Control
PLC Programmable Logic Controller
PMS Performances Measurement Systems
SCADA Supervisory Control And Data Acquisition
SOAP Simple Object Access Protocol
UML Unified Modeling Language
XML eXtensible Markup Language

REFERENCES

Chiron, F., 2009. Contribution à la flexibilité et à la rapidité de conception des systèmes automatisées avec l'utilisation d'UML. Thèse de Doctorat en Informatique ; université Blaise Pascal – Clermont

Curtis, D., Stone, C., Bradley, M., 2009. IIOP: OMG's Internet Inter-ORB Protocol A Brief Description. Object Management Group (OMG). Available from: http://www.omg.org. [accessed 10 May 2010]

Halang, W. A., Sanz, R., Babuska, R., Roth, H., 2006. Information and communication technology embraces control. Status report prepared by the IFAC: Coordinating Committee on Computers, Cognition and Communication. Annual Reviews in Control 30 31–400

ISA. 2000. ANSI/ISA 95.00 03-2000. Enterprise/Control System Integration - Part 3: Activity Models of Manufacturing Operations Management. The Instrumentation, Systems, and Automation Society.

Kettani. N., D. Mignet, P. Paré, C. Rosenthal-Sabroux, 1999. De Merise à UML. Editions Eyrolles.

Kouiss, K., Chiron, F., David, S., Thomas, M., 2006. A Component based development of user manufacturing systems. IDMME 2006. Grenoble France.

Krassi. B. A., 2002. Distributed computing and automation: Industrial applications. URL : http://www.automationit.hut.fi/julkaisut/documents/seminars/sem_s02/Krassi_paper.pdf.

MESA International. 2000. "Control definition and MES to control data flow possibilities". White paper. N :3 pp :1-7. http://www.mesa.org.

OPC Task Force., 1998. OPC Overview. Version 1.0. OPC Foundation. Available from: www.opc.org. [accessed 10 May 2010]

Yang. C. O., Guan, T. Y., Lin, J. S., 2000. Developing a computer shop floor control model for CIM systems – using object modelling technique, Computers in Industry 41. pp: 213-238.