RFID-Based Manufacturing Monitoring and Analysis System

Sergei Drannikov

Submitted to the Graduate School of Engineering and Natural Sciences in partial fulfillment of the requirements for the degree of Master of Science

Sabanci University
Autumn 2007

APPROVED BY

Assist. Prof. Dr. Hüsnü Yenigün .....
(Thesis Supervisor)

Assoc. Prof. Dr. Bülent Çatay .....
(Thesis Co-Supervisor)

Assoc. Prof. Dr. Erhan Budak .....

Assist. Prof. Dr. Tonguç Ünlüyurt .....

Assist. Prof. Dr. Ayhan Bozkurt .....

Dedicated to

My wife Irina, my daughter Alina and the beautiful Siberian city of Krasnoyarsk

RFID-Based Manufacturing Monitoring and Analysis System

by Sergei Drannikov

Abstract

Radio Frequency Identification (RFID) has become an important driver in the production and logistics activities of today's information-based industries and economies. With the innovative developments in information and communication technologies, companies focus more on how these changes can be implemented and promoted in order to benefit from these technologies and to improve their value-added processes. This thesis presents an RFID-based manufacturing monitoring and analysis system to bridge the gap between the physical flow of materials on the shop floor and manufacturing information and execution systems by allowing to quickly model, develop, deploy, monitor, and analyze a discrete manufacturing environment. The system was designed to be as generic as possible so that it can be applied to any existing manufacturing process without or with very little modification. In this way our system may be considered what is called an RFID-enabling technology.

RFT Tabanlı İmalat İzleme ve Analiz Sistemi

Sergei Drannikov

Özet

Radyo Frekans Tanıma teknolojisi, bugünkü bilgi tabanlı endüstri ve ekonomilerdeki üretim ve lojistik faaliyetlerinin önemli bir faktörü olmuştu. Bilgi ve iletişim teknolojilerindeki yenilikçi gelişmelerle birlikte şirketler daha çok, bu teknolojilerden yararlanmak ve katma değer süreçlerini ilerletmek için bu değişikliklerin nasıl uygulanacağına ve terfi edileceğne odaklanıyor. Bu tez, ayrık üretim çevresinin çabuk modellenmesini, geliştirilmesini, uygulanmasını, izlenmesini ve analiz edilmesini sağlayarak, üretim bölümündeki malzemelerin fiziksel akışıyla üretim bilgi ve işletme sistemleri arasındaki boşluğu kapatacak RFT tabanlı üretim izleme ve analiz sistemini sunuyor. Sistem, hiç değişiklik yapılmadan veya az bir değişiklikle varolan üretim süreçlerine uygulanabilsin diye mümkün olduğu kadar jenerik tasarlanmıştır. Bu bakımdan sistemimiz, RFT etkinleştiren teknoloji olarak da nitelendirilebilir.

Acknowledgements

My sincerest thanks go to Assist. Prof. Dr. Hüsnü Yenigün, Assoc. Prof. Dr. Bülent Çatay, Assoc. Prof. Dr. İbrahim Tekin and Assoc. Prof. Dr. Erhan Budak for their dedication to our project and patience in assisting me with this thesis. I appreciate their valuable advice and efforts offered during the course of my studies.

I would also like to thank my jury members, Assist. Prof. Dr. Tonguç Ünlüyurt and Assist. Prof. Dr. Ayhan Bozkurt, for their equally valuable support generously given during the writing of my thesis.

I would also like to mention that my graduate work is supported by the Scientific and Technological Research Council of Turkey (TÜBİTAK).

I would also like to express my gratitude to Teksan Generator Factory staff and especially to Mr. Faruk Sezen and Mr. Yunus Teksan for their invaluable help and support provided during the work.

Special thanks go to my friends Mehmet Abbak, Hüseyin Ergün, Serhat Alagöz, Ergi Şener, Sefa Özbek and Ercan Kaymaksüt. I appreciate their friendship and sympathetic help which made my life easier and more pleasant during graduate studies.

Especially, I would like to give my special thanks to my wife Irina whose patient love enabled me to complete this work.

Lastly, I would like to thank my parents for their enormous encouragement and assistance, for without them, this work would not have been possible.

Contents

Abstract **iv**

Özet **v**

Acknowledgements **vi**

1 Introduction **1**

1.1 Overview of Computer-Integrated Manufacturing 2

1.2 Overview of RFID Technology 4

2 Relevant Literature **8**

3 RFID-Based Manufacturing Monitoring and Analysis System **13**

3.1 Overview 13

3.2 Setup and Configuration Studio (RFPML Studio) 16

3.3 Process Description Formats 17

3.3.1 Business Process Management (BPM) 19

3.3.2 XML Process Definition Language (XPDL) 20

3.3.3 Process Specification Language (PSL) 20

3.3.4 Sensor Markup Language (SensorML) 21

3.3.5 RFPML 21

3.4 Runtime 26

3.4.1 Client Software 26

3.4.2 Server Software 27

3.4.3 Database Structure 30

3.4.4 Client and Server configuration files 31

3.4.5 Client-Server Interaction Protocol 33
3.4.6 Client-Server Interaction Algorithm 37
3.4.7 RFPML Web Monitoring Tool 39
3.5 Implementation 39
3.5.1 RFPML Studio 39
3.5.2 Runtime 41

4 Factory Integration **42**

5 Conclusions and Future Work **47**

Appendix **48**

A Descriptions and XML Schemas of some of the formats developed for the system **48**
A.1 RFPML Document Format 48
A.2 Client Request Message Format for the Client-Server Interaction Protocol 54

B Examples files for some of XML formats developed for MaMAS Runtime **56**
B.1 Example of a Client Configuration File 56
B.2 Example of a Server Configuration File 57

Bibliography **58**

List of Figures

1.1 The diagram shows a typical backscatter scheme for RFID tags, which are powered using the energy contained in the requesting wave from the reader device. 5

3.1 Major parts of the system 16
3.2 RFPML Studio screenshot 17
3.3 RFPML Studio screenshot, showing property editor for components . 18
3.4 Process View elements 22
3.5 RFID View elements 23
3.6 The Process View of an example production process 25
3.7 The RFID View of an example production process 26
3.8 Example Windows Forms .NET client 28
3.9 Example Windows CE.NET client 29
3.10 Hierarchical structure of workstations for the example sequence of workstations 30
3.11 Database Structure 31
3.12 A sample client configuration file saving dialog 33
3.13 A sample server configuration file saving dialog 34
3.14 Example of an XML-based client request file 35
3.15 Client-Server Interaction Algorithm 38
3.16 RFPML Web Monitoring Tool 40

4.1 Production flow for a generator. Courtesy of Teksan Generator Factory. 45
4.2 A Full View of the generator production process of the Teksan factory 46
4.3 Tag event data table obtained for 3 generator assembly process runs . 46

Chapter 1

Introduction

Most modern manufacturing management initiatives are real-time driven and based on information from events as they occur [1]. Monitoring the part flows in a manufacturing system time-wise and location-wise is very critical for productivity, cost, quality, inventory, and speed. In companies where the production automation has been implemented the parts are monitored by the system anyway; or barcode readers are used to obtain that information. The barcode systems require either full operator support or the part has to be presented to the reader with an appropriate orientation and proximity. This increases the positioning and the reading time. Intelligently networked control systems equipped with RFID technology, on the other hand, may allow better monitoring of the manufacturing processes with expedient material flow and more effective planning and control.

This research is motivated by the need of bridging the gap between the physical flow of components/products and the corresponding information flow in conventional manufacturing systems. RFID technology can provide the solution for bridging and even closing the gap [2]. Our aim is to develop an RFID-based manufacturing monitoring and analysis system to monitor where and when a part is located throughout its flow in a production process using RFID tags placed on the parts and antennas located at appropriate places. This system will enable the visibility of multiple parts simultaneously without requiring particular positioning of the part and minimizing operator support to the possible extent. Moreover, relevant product and production information may be recorded on the tags as well, if necessary. Since the

process history of each part will be monitored in real-time using the information saved either on the tag or in the database potential future mistakes which may occur in the subsequent production, assembly, or service steps may be avoided. The proposed monitoring system will provide similar abilities of those of a computer integrated manufacturing (CIM) system. However, CIM is very rarely used in Small and Medium Enterprises (SME) because of high investment and maintenance costs. In this regard, our system may be an alternative as an economical and efficient method of production monitoring in SMEs as well as in large enterprises.

The remainder of the thesis is organized as follows: below is a brief overview of CIM and RFID technologies. Chapter 2 reviews the literature on RFID-enabled technologies and applications. Chapter 3 describes the proposed RFID-based manufacturing monitoring system, and some of its implementation details are given as well. Factory integration is discussed in Chapter 4. Finally, Chapter 5 presents the implications and concluding remarks and provides future research directions.

1.1 Overview of Computer-Integrated Manufacturing

The Computer and Automation Systems Association (CASA) of the Society of Manufacturing Engineers (SME) defined *CIM* as follows:

CIM is the integration of the total manufacturing enterprise through the use of integrated systems and data communications coupled with new managerial philosophies that improve organizational and personnel efficiency. [3]

CIM describes a new manufacturing paradigm in which speaking simplistically the entire production process is controlled by a computer. CIM systems can include many advanced manufacturing technologies (sometimes called *subsystems* of CIM), among others:

* Robotics
* CAD/CAM, Computer-Aided Design/Computer-Aided Manufacturing
* CAPP, Computer-Aided Process Planning

* CAQ, Computer-Aided Quality Assurance
* ERP, Enterprise Resource Planning
* CNC, Computer Numerical Control machine tools
* DNC, Direct Numerical Control machine tools
* ASRS, Automated Storage and Retrieval Systems
* AGV, Automated Guided Vehicles
* Automated Conveyance Systems
* FMS, Flexible Manufacturing Systems
* Lean Manufacturing
* Cellular Manufacturing
* JIT, Just-In-Time production
* A business system integrated by a common database.
* Office automation

As can also be seen from the above list, among the important building blocks which constitute a successful CIM system the following can be emphasized: a commitment to total enterprise quality, continuous improvement, customer satisfaction, use of a single computer database for all product information that is the basis for manufacturing and production decisions in every department, removal of communication barriers among all departments, and the integration of enterprise resources. [4] [5] [6]

According to another definition, CIM is a process of using computers and communication networks to transform islands of enabling technologies into a highly interconnected manufacturing system. CIM involves integration of advanced technologies in various functional units of an enterprise in an effective manner to achieve the corporate objective of the manufacturing enterprise.

The following key challenges are encountered when implementing CIM systems:

- *Integration of components from different suppliers*: When different machines, such as CNC, conveyors and robots, are using different communications protocols. In the case of AGVs, even differing lengths of time for charging the batteries may cause problems.

- *Data integrity*: The higher the degree of automation, the more critical is the integrity of the data used to control the machines. While the CIM system saves on labor of operating the machines, it requires extra human labor in ensuring that there are proper safeguards for the data signals that are used to control the machines.

- *Process control*: Computers may be used to assist the human operators of the manufacturing facility, but there must always be a competent engineer on hand to handle circumstances which could not be foreseen by the designers of the control software. [6]

As a final note in this brief introduction, the *M* in CIM is rather misleading because the word *manufacturing* makes it easy to assume that it is intended for use only in manufacturing or for integrating the operation of shop floor machinery. The *M* in CIM is actually the connection of all manufacturing-related functions linked in a computer network in an integrated manner. Production planning and scheduling, production control, process design, and purchasing, as well as all manufacturing processes and equipment, must be integrated in a total CIM system. [7]

1.2 Overview of RFID Technology

Radio-frequency identification (RFID) is an automatic identification method, relying on storing and remotely retrieving data using devices called RFID tags or transponders. [8]

An RFID tag in the Figure 1.1 is an object that can be applied to or incorporated into a product, animal, or person for the purpose of identification using radiowaves. Some tags can be read from several meters away and beyond the line of sight of the reader.

Most RFID tags contain at least two parts. One is an integrated circuit for storing and processing information, modulating and demodulating a (RF) signal and can also be used for other specialized functions. The second is an antenna for receiving and transmitting the signal.

RFID tags come in three general varieties: passive, active, or semi-passive (also known as battery-assisted). Passive tags require no internal power source, thus being pure passive devices (they are only active when a reader is nearby to power them), whereas semi-passive and active tags require a power source, usually a small battery.

To communicate, tags respond to queries generating signals that must not create interference with the readers, as arriving signals can be very weak and must be told apart. Besides backscattering, load modulation techniques can be used to manipulate the reader's field. Typically, backscatter is used in the far field, whereas load modulation applies in the nearfield, within a few wavelengths from the reader.

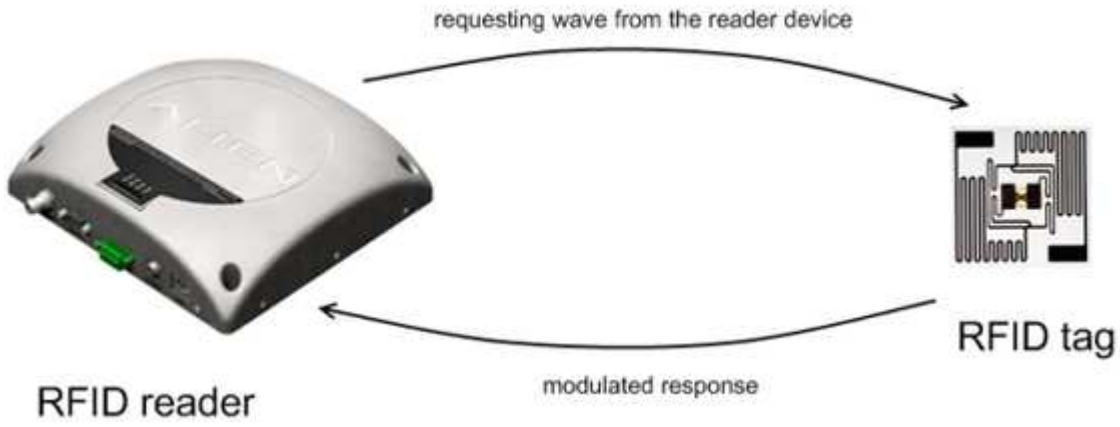

Figure 1.1: The diagram shows a typical backscatter scheme for RFID tags, which are powered using the energy contained in the requesting wave from the reader device.

Passive RFID tags have no internal power supply. The minute electrical current induced in the antenna by the incoming radio frequency signal provides just enough power for the Complementary metaloxidesemiconductor (CMOS) integrated circuit in the tag to power up and transmit a response. Most passive tags signal by backscattering the carrier wave from the reader. This means that the antenna has to

be designed both to collect power from the incoming signal and also to transmit the outbound backscatter signal. The response of a passive RFID tag is not necessarily just an ID number; the tag chip can contain non-volatile, possibly writable Electrically Erasable Programmable Read-Only Memory (EEPROM) for storing data.

Unlike passive RFID tags, active RFID tags have their own internal power source, which is used to power the integrated circuits and broadcast the signal to the reader. Active tags are typically much more reliable (i.e. fewer errors) than passive tags due to the ability for active tags to conduct a "session" with a reader. Active tags, due to their onboard power supply, also transmit at higher power levels than passive tags, allowing them to be more effective in "RF challenged" environments like water (including humans/cattle, which are mostly water), metal (shipping containers, vehicles), or at longer distances, generating strong responses from weak requests (as opposed to passive tags, which work the other way around). In turn, they are generally bigger and more expensive to manufacture, and their potential shelf life is much shorter.

Many active tags today have practical ranges of hundreds of meters, and a battery life of up to 10 years. Some active RFID tags include sensors such as temperature logging which have been used to monitor the temperature of perishable goods like fresh produce or certain pharmaceutical products [9]. Other sensors that have been married with active RFID include humidity, shock/vibration, light, radiation, temperature, and atmospherics like ethylene [10]. Active tags typically have much longer range (approximately 500 m/1500 feet) and larger memories than passive tags, as well as the ability to store additional information sent by the transceiver.

Semi-passive tags are similar to active tags in that they have their own power source, but the battery only powers the microchip and does not broadcast a signal. The RF energy is reflected back to the reader like a passive tag. An alternative use for the battery is to store energy from the reader to emit a response in the future, usually by means of backscattering.

The battery-assisted receive circuitry of semi-passive tags lead to greater sensitivity than passive tags, typically 100 times more. The enhanced sensitivity can be leveraged as increased range (by a factor 10) and/or as enhanced read reliability (by

one standard deviation).

Semi-passive tags have three main advantages 1) Greater sensitivity than passive tags 2) Better battery life than active tags. 3) Can perform active functions (such as temperature logging) under its own power, even when no reader is present. The antenna used for an RFID tag is affected by the intended application and the frequency of operation. Low-frequency (LF) passive tags are normally inductively coupled, and because the voltage induced is proportional to frequency, many coil turns are needed to produce enough voltage to operate an integrated circuit.

At 13.56 MHz (High frequency or HF), a planar spiral with 57 turns over a credit-card-sized form factor can be used to provide ranges of tens of centimeters. These coils are less costly to produce than LF coils, since they can be made using lithographic techniques rather than by wire winding, but two metal layers and an insulator layer are needed to allow for the crossover connection from the outermost layer to the inside of the spiral where the integrated circuit and resonance capacitor are located.

Ultra-high frequency (UHF) and microwave passive tags are usually radiatively-coupled to the reader antenna and can employ conventional dipole-like antennas. Only one metal layer is required, reducing cost of manufacturing.

HF and UHF tag antennas are usually fabricated from copper or aluminum. Conductive inks have seen some use in tag antennas but have encountered problems with IC adhesion and environmental stability. [11]

Chapter 2

Relevant Literature

Different uses of the RFID technology were reported in recent years in the manufacturing industry. Ford Motor Company has successfully implemented an RFID-based Just-In-Time (JIT) manufacturing model at its facility in Cuautitlan, Mexico [12]. In the manual coding system, the identification sheets were manually updated at every stage in the production line. In the RFID-based system, however, updates are automatically written on the tag as the vehicle advances on the production line without the risk of operator error.

IBM has transformed chip production at its Fishkill plant with a semiconductor manufacturing system that leverages real-time information to automatically control the fabrication process, enabling employees to work more productively and be more responsive to customers product status inquiries [13]. IBM has accomplished this using IBM SiView Standard, a manufacturing execution system that the company integrated with its own wireless e-business technology. SiView Standard leverages information from IBM DB2 Universal Database to automatically control each step of the fabrication process. DB2 manages information about the fabrication processes that need to be applied to every wafer containing chips, and supports data analysis tools that provide production-related statistics. IBM WebSphere MQ provides the messaging platform that enables DB2 to exchange information with the production tools and other application programs used to run the plant.

BMW and Vauxhall use RFID tags to enable accurate customization of customer orders [14]. A read/write smart tag is programmed in the customer order. The tag

is then attached to and travels with the car during the production process. This tracking ensures that the car is manufactured with the correct color, model, interior, and any other option the customer specifies.

Extensive studies were also performed on the various types of systems that RFID technology can be used in. Zhou et al. [15] discuss an architecture where communication between RFID-based data acquisition system and various monitoring terminals is performed using RFID, Bluetooth and Internet channels. Koumpis et al. [16] address various architectures related to wireless/wired communications in manufacturing environments. Brewer et al. [17] propose "Intelligent Tracking Technologies", or IT2, which comprises Global Positioning Systems (GPS), Geographic Information Systems (GIS), wireless communications, and RFID to enable dynamic scheduling in manufacturing and supply chain management. Huang et al. in [18] and [19] develop "Wireless Manufacturing" (WM) technology to manage job shop Work-In-Progress (WIP) inventories in real time. The emphasis is placed upon how to avoid changing from functional (or "walking-worker fixed-position" in their terms) to cellular layouts "in order to retain existing operational flexibility while improving efficiency and capacity".

By taking advantage of data capacity stored in an RFID tag, critical manufacturing information on a product can be locally stored with the product. Such a feature may be very important when it is not possible to work in a networked environment. Qiu [20] offers a framework to enable the instant delivery of pertinent data and information on a uniquely identifiable job/product at point-of-need across factories.

Alternatively, Yagi et al. [21] discusses the application of RFID in construction production field where information related to a product is carried by the product itself and can be handled to manage the whole system.

The interested reader is referred to [22] and [23] for more examples and case studies on how RFID technology was successfully implemented in real-world manufacturing projects.

In the field of manufacturing monitoring and control systems a great amount of research is conducted on the use of so called software agents. In computer science,

a software agent is a piece of software that acts for a user or other program in a relationship of agency. Such "action on behalf of" implies the authority to decide which (and if) action is appropriate. The idea is that agents are not strictly invoked for a task, but activate themselves. [24]

Tnazefti-Kerkeni et al. [25] presents a multi-blackboard approach to design and implement a control/monitoring system for the Automation of Production Systems. The proposed architecture is composed of several control/monitoring agents (CMAs) organized hierarchically. Communication between agents is done through a hierarchy of blackboards, where consistent replicated data is kept.

Sauer and Sutschet [26] developed Provis.Agent - the first agent-based production monitoring and control system for distributed real-time production monitoring that allows integration with other shop-floor related applications.

Leitao and Restivo [27] intends to introduce an agent-based approach to the manufacturing problem, that uses holonic concepts, is focused on distributed manufacturing shop floor control for discrete batch production, considers the optimization of set-up and maintenance operations, and develops mechanisms for agile and fast reaction to disturbances without compromising the global production optimization. "Holonic" here refers to systems (holons) which are parts of some other, larger and more complex systems and at the same time are autonomous self-reliant units, which have a degree of independence and handle contingencies without asking higher authorities for instructions [28].

An interesting case (not software agent-based) is presented by Yurtsever and Pierce [29]. They developed and implemented the so called Graphical Manufacturing Monitoring System (GraMMS), consisting of 4 main applications namely; Dynamic Dispatch, WIP Monitoring System (WMS), Equipment Management System (EMS), and Throughput Monitoring System (TMS).

Up to here cited works are in essence "applications" (as opposed to frameworks). The evident shortcoming of them is that they are applicable only to concrete projects and situations, that is they lack any generality at all. Our work on the other hand is the development of an enabling technology, i. e. the technology which enables existing manufacturing processes with monitoring and control means through using

the RFID technology. Our aim is not only developing of an RFID enabled monitoring system at a given shop floor, but we want to develop a system that would simplify and accelerate the application of RFID enabled monitoring systems at any given factory.

The only example of an RFID enabling technology we could find in the literature was one of IBM's alphaWorks projects namely the RFID Integrated Solution Enablement (RISE) [30]. As is written on its website, *RISE is a model-driven solution integration framework for RFID solutions. RISE facilitates the following:*

* *rapid creation of solutions through GUI-based component composition and software reuse*

* *effective testing with simulation environment that allows developers to validate the logical flow of the solution prior to real deployment into physical environment*

* *easy solution deployment and management through polymorphism and a standard method of application lifecycle management (OSGi).*

RISE also provides a set of built-in library components, which vary from four different vendor models of RFID readers to some logical components such as a duplicate tag read filter.

Some of the similarities between RISE and our system:

- Both systems have to a great extent similar overall component-based architectures. Models are produced in a modeling environment, then solutions are deployed in a runtime execution platform.

- Both RISE and our system provides GUI-based modeling environments for solutions prior to deployment. RISE has developed an Eclipse-based plugin for this purpose, and we developed a .NET-based model creation and management tool called RFPML Studio. RFPML stands for Radio Frequency Process Modeling Language and is explained in the next Chapter.

- RISE persists solution models in XML-based RISE model format. RFPML Studio also persists models in XML-based format (with a different schema of course).

And some of the differences:

- RISE uses a standard Java-based method of application lifecycle management called Open Services Gateway initiative (OSGi) for solution deployment and configuration management. Deployment of models in our system occurs through the GUI-based modeling environment.

- RISE uses a third-party execution engine called called Ptolemy II developed and supported by the UC Berkeley [31]. Our system only needs .NET Framework 2.0 on any supporting Windows Operating System.

- RISE provides a testing with simulation environment, our system does not.

- RISE does not have any monitoring and analysis module, our system has a web-based monitoring tool developed by a group of students of the Sabanci University [32].

- RISE has several prerequisites some of which are quite expensive. Some prerequisites (by IBM) are projects either discontinued by IBM or transformed into some other projects. So, as of January 2008, it is impossible to install RISE on a computer. Our system does not have any particular prerequisites except for the .NET Framework 2.0 mentioned above.

Chapter 3

RFID-Based Manufacturing Monitoring and Analysis System

3.1 Overview

The ability to monitor manufacturing parts and processes in real-time can provide many benefits from the point of view of production planning and productivity. We list below some problems and the capabilities of our proposed RFID-based manufacturing monitoring and analysis system to overcome these problems.

i) In order for the right parts to go to the right workstations at the right time the parts need to be tracked in real-time. With the RFID-enabled system the arrival and departure times of a part in a workstation can be recorded automatically. An integrated planning and/or monitoring software such as Enterprise Resource Planning (ERP) systems may access and utilize this information in real-time.

ii) Process flow and time analyses should be performed very carefully and can be time consuming. If such analyses are not performed or performed improperly it becomes difficult to trace workstation related problems in the production processes. In the proposed system since the workstation each part visits in the process flow is traced in real-time, such problems will be fairly easy to detect and resolve.

iii) Work-in-process (WIP) stocks are critical for productivity, bottleneck management, and inventory control. Since the RFID-based system provides visibility at each stage the location and quantity of the parts can be monitored in real-time and the arising bottlenecks can be detected and resolved more rapidly.

iv) Even simple parts in manufacturing processes can consist of many different sub-assemblies and components. Different problems can arise in the flow of these items during the production process. The proposed RFID system provides the capability of detecting such problems faster and taking necessary actions properly.

In the proposed RFID-based manufacturing monitoring and analysis system we collect the following information related to a part and write it to a database:

a) Arrival time to a workstation

b) Departure time from the workstation

c) Some extra information which may be relevant to the outcome of the operation performed on the workstation.

Note: We actually distinguish between two types of workstations. One of them records both the arrival and the departure times and the other one records only the departure time.

A typical RFID system consists of the following three components:

(i) Tags attached to the physical objects to be identified

(ii) Readers, and

(iii) A host system that contains information on the identified object and distributes information to other remote data processing systems, such as ERP, Manufacturing Execution Systems (MES), etc. [2]. The host system can be used for other purposes as well, such as RFID infrastructure management and configuration, data storage configuration, integration to other enterprise subsystems, etc. The tag data can also be stored on tags themselves instead of the host system, e.g., see [20].

An essential part of our research is the development of such a host system. The system needs to be as generic as possible, such that it can be easily adapted to most manufacturing environments with some setup and configuration options. In what follows we discuss the developed host system.

Typically development of custom solutions with the help of our system will consist of 2 stages:

1) Model development. By models we mean constructs used to represent the logical flow of a manufacturing process. We developed a special structure for the models, which we called RF Process Modeling Language, or RFPML. On top of this we built a Graphical User Interface tool in order to create, manage and deploy the models, which we called RFPML Studio. Models include the description of RFID infrastructure underlying the system and to some extent they also include the specification of the way the data would be collected (the stations from which the data will be collected, type of workstations regarding the data collection pattern (arrival/departure or just departure)).

2) Model deployment. We also created runtime environment which is used to do actual job of monitoring and control. Runtime consists of server-side part and client-side components (they may also be called edge components). Runtime uses the developed models in order to properly monitor and if necessary control the flow of manufacturing parts. Server software handles requests from clients, performs all database queries, and manages RFID hardware if necessary. Both client and server parts of the runtime use model information in the form of initialization files and database entries generated from the model. Such generation (what we call code generation) occurs on the RFPML Studio's side.

Therefore our system mainly consists of 3 primary parts: RFPML Studio, MaMAS (stands for **Ma**nufacturing **M**onitoring and **A**nalysis **S**ystem) Runtime, and also Web Monitoring tool. The third part comes from the work performed by a group of students from the Sabanci University who developed a web-based monitoring software for the system [32]. Using the generated RFPML document the

tool reconstructs the graphical layout, then uses specially devised custom queries to extract the system's runtime information from the database.

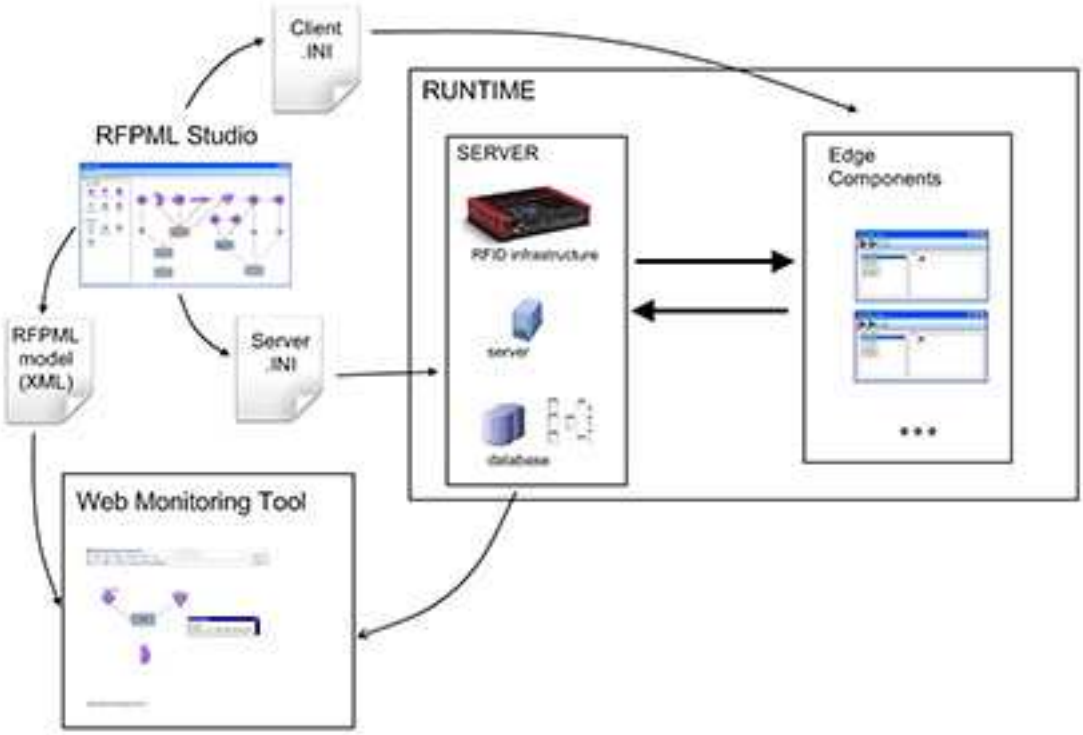

Figure 3.1: Major parts of the system

3.2 Setup and Configuration Studio (RFPML Studio)

The setup and configuration part of the system is implemented as a graphical modeling environment as depicted in Fig. 3.3. The environment has all the basic necessary user interface (UI) tools for the end user to construct a model of his/her specific RFID-enabled solution appropriately.

Visually the environment consists of three major parts:

1. Left pane containing two tabs:

    a. A library of the building blocks of an RFID-enabled manufacturing process, naturally grouped into two categories: Process components, and RFID infrastructure components.

b. A property grid to edit the properties of the building blocks

3. The working area at the center of the screen, where all the actual modeling work happens.

4. The toolbar providing some more common and frequently performed functions.

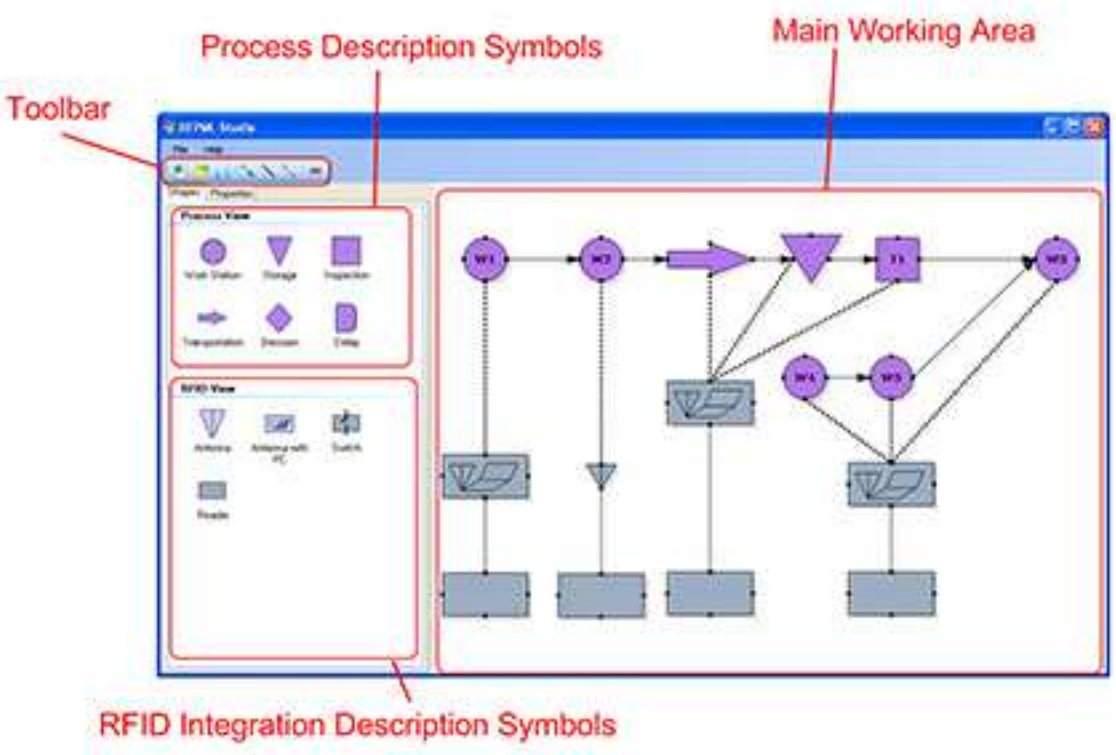

Figure 3.2: RFPML Studio screenshot

Users drag components from the library tab and drop them on the working area. Then they can adjust their positions, connect them with one another to construct a process flow, set their properties using the Properties pane, and perform some other basic modeling functions.

3.3 Process Description Formats

At the end the user can save the work to an XML-like format. The purpose of this is two-fold:

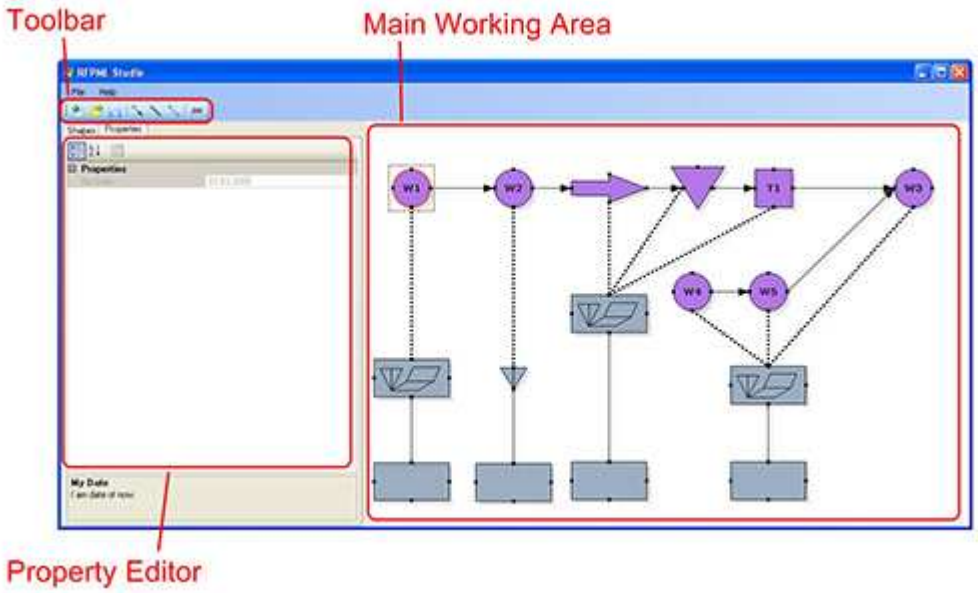

Figure 3.3: RFPML Studio screenshot, showing property editor for components

(i) persistence means, that is to be able to work with the models later and

(ii) it will be used in generating software components necessary for specific applications.

In choosing the format, we conducted a thorough examination of the literature and the Internet on the subject of manufacturing process XML specifications. We had in mind the following criteria when searching:

* It should be able to represent manufacturing processes.

* It should be extensible to represent RFID infrastructure, such as readers, antenna, etc.

* It should have graphical notation capabilities or at least be easily extensible in this respect, i. e. we should at least be able to represent the entire XML document using some graphical elements.

Among others, we studied Business Process Management (BPM) together with some related technologies, such as Business Process Management Notation (BPMN), Business Process Execution Language (BPEL) with variants/successors; and also XML

Process Definition Language (XPDL), Process Specification Language (PSL) and SensorML.

3.3.1 Business Process Management (BPM)

Business Process Management (BPM), like Service Oriented Architecture (SOA) is one of those "buzz enterprise technologies" of recent years plenty of resources talk about but few seem to properly implement at their own enterprises. The definition of BPM by the Association of Business Process Management Professionals (ABPMP):

Business Process Management (BPM) is a disciplined approach to identify, design, execute, document, monitor, control, and measure both automated and non-automated business processes to achieve consistent, targeted results consistent with an organization's strategic goals. BPM involves the deliberate, collaborative and increasingly technology-aided definition, improvement, innovation, and management of end-to-end business processes that drive business results, create value, and enable an organization to meet its business objectives with more agility. [33]

As seen from the above BPM is a fairly general concept. That's why academic authors may give different meaning to this concept and vendors generally provide varying implementations. There are also many technologies related in some way to BPM:

- Business Process Modeling Notation (BPMN). Its main goal is to provide a standardized graphical notation for business processes as it is evident from the name. One of the reasons for introducing the concept of BPM in the technological aspect is that it will provide a general standard notation platform for business analysts who create the business processes, technical developers who implement them and managers who monitor and manage them. That's why the notational aspect of BPM quickly evolved into separate standard developed by the Business Process Management Initiative (BPMI).

  Business Process Execution Language (BPEL). As it is again evident from the name BPEL is a business process language that is executable. Because of its implied practicality it was widely embraced by the industry from its very

beginning.

For our purposes BPM is too general. We needed something narrower for both aspects of our system, manufacturing and RFID infrastructure.

3.3.2 XML Process Definition Language (XPDL)

XPDL is also related to Business Process Management. As described on the website of the Workflow Management Coalition - the organization which created and maintains XPDL:

The goal of XPDL is to store and exchange the process diagram, to allow one tool to model a process diagram, and another to read the diagram and edit, another to "run" the process model on an XPDL-compliant BPM engine, and so on. For this reason, XPDL is not an executable programming language like BPEL, but rather a process design format that literally represents the "drawing" of the process definition. Specifically, it has 'XY' or vector coordinates, including lines and points that define process flows. This allows an XPDL to store a one-to-one representation of a BPMN process diagram. For this reason, XPDL is effectively the file format or "serialization" of BPMN, as well as any non-BPMN design method or process model which use in their underlying definition the XPDL meta-model. [34]

As can be seen from the above XPDL emphasizes yet another aspect of business processes which we also needed for our system - the one-to-one serialization from the graphical format to the textual format of a process.

The problem with XPDL as regarding applicability to our system is actually the same as with the BPM - it is far too broad, we needed something more specific.

3.3.3 Process Specification Language (PSL)

PSL defines a neutral representation for manufacturing processes. Process data is used throughout the life cycle of a product, from early indications of manufacturing process flagged during design, through process planning, validation, production scheduling and control. In addition, the notion of process also underlies the entire manufacturing cycle, coordinating the workflow within engineering and shop floor

manufacturing. [35]

Unfortunately, PSL lacked the necessary degree of extensibility to provide for the RFID infrastructure of our system and was fairly difficult to implement in graphical respect.

3.3.4 Sensor Markup Language (SensorML)

SensorML provides standard models and an XML encoding for describing any process, including the process of measurement by sensors and instructions for deriving higher-level information from observations. Processes described in SensorML are discoverable and executable. All processes define their inputs, outputs, parameters, and method, as well as provide relevant metadata. SensorML models detectors and sensors as processes that convert real phenomena to data. [36]

SensorML on the other hand almost totally lacked the ability to represent adequately manufacturing processes.

3.3.5 RFPML

So we decided to implement our own format which we called RFID Process Markup Language (RFPML). In this section we briefly describe the language.

An RFPML document has 2 equivalent notations: a graphical notation and an XML based textual notation. We firstly describe its graphical components. An RFPML document has 2 major parts: **Process Description Symbols** and **RFID Integration Description Symbols**. The Process Description Symbols describe elements related to the manufacturing process itself, while the RFID Integration Description Symbols describe elements related to the corresponding RFID infrastructure.

Process Description Symbols

In the Process View, we decided to utilize 5 symbols from the production charts symbol set standardized by the American Society of Mechanical Engineers (ASME) [37]: Workstation, Storage, Inspection, Delay, and Transportation. We have also

added the Decision element to support generic flowcharting notations. The symbols are illustrated in the figure below.

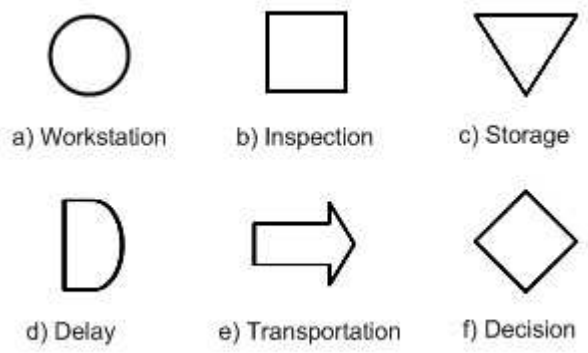

Figure 3.4: Process View elements

Here is a concise description of each symbol:

a) *Workstation.* Also called an *Operation.* An operation occurs when an object is arranged or prepared for another step, assembled or disassembled or intentionally changed.

b) *Inspection.* An inspection occurs when an object is verified for quality or quantity in any of its characteristics.

c) *Storage.* A storage occurs when an object is kept and protected against unauthorized removal.

d) *Delay.* A delay occurs when an object waits for the next planned action.

e) *Transportation.* A transportation occurs when an object is moved from one location to another.

f) *Decision.* A decision or branching point. Lines representing different decisions emerge from different points of the diamond. [38]

RFID Integration Description Symbols

RFID View helps process designer to define precisely the underlying RFID infrastructure used to collect the data. In other words, the designer will be able to define

which workstation will be collecting which type of data, in what formats, from which readers, etc. For visual representation of these specific features of our system we propose the notation of symbols depicted in Figure 3.5.

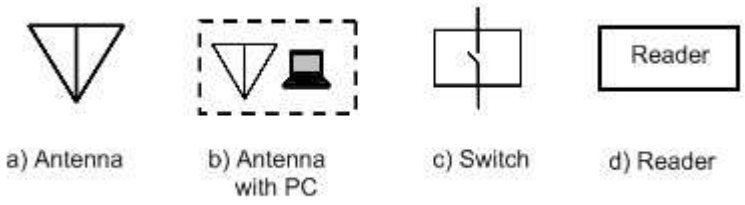

Figure 3.5: RFID View elements

Again a brief description of each symbol is as follows:

a) *Antenna*. We decided to use this symbol since it already has a widespread usage in the literature. An antenna will be connected from one end to a reader (either directly or through a switch) and from the other to the element of a manufacturing process, e. g. a workstation (for an example see the antenna symbol connecting a Workstation named "W2" with a Reader on Figure 3.3). A link between the workstation and an antenna means that during the operation of the system information about the workstation will be collected. The information will be in the form of arrival and departure times, or just departure times (depending on the workstation type) of a part to/from the workstation. If some extra information is needed we will have to use another element, which we called Antenna with PC. For example, we may need to be able to record the result of an operation on an Inspection element. In this case an operator has to enter the result into the system. For this type of operations we need a PC attached to the antenna. That's the reason for introducing the next type of symbol.

b) *Antenna with PC*. As described above we need a PC for operations requiring manual data entry from operators (see, for example, an Antenna with PC connecting "T1" station with a Switch on Figure 3.3). We need the kind of antenna-PC pair for other situations as well. For example, when a process de-

signer decides to have multiple workstations attached to the same antenna we need a manual entry from the operator in order to find out on which workstation the tagged part under consideration is being processed (see Workstations "W3" and "W5" connected to a Reader through an Antenna with PC on Figure 3.3 for an example). Actually this element was designed bearing in mind the probability of the following real-life scenario. Process designer may want to be able to collect data from more than one workstation using hand-held RFID reader. In this case it will inevitably be asked from the operator which workstation s(he) currently is at.

Since Antenna with PC will definitely require operator support it should not be used for workstations with automatic data collection. In this situation we need to use the Antenna symbol.

Antenna with PC will be connected either directly to an RFID reader or through a switch. From the other side this symbol will be connected to one or more manufacturing process description symbols. So it will be clear that for workstations connect to the Antenna with PC manual data entry will be required.

The PC part of the Antenna with PC element can be a desktop computer, a notebook or a palm computer. The type of the PC can be described in the form of a property of this symbol. In any case this computer will be connected to a LAN.

c) *Switch.* In our system switches are used for connecting multiple antennas to a single reader. So the information such as which antenna connected to which reader through which switch is represented graphically. More than one Antenna or Antenna with PC symbol will be connected to a Switch symbol. The control of switches will be done through a server computer.

d) *Reader.* It illustrates RFID readers as attached to antennas either directly or through a switch. The readers can be connected either to a LAN or directly to a PC through an RS232 port.

Elements are combined to each other using different types of connections. Process Description Symbols connect to each other using normal connections having starting cap of the line flat and ending cap - an arrow. Lines connecting Process Description Symbols to RFID Integration Description Symbols are dashed and have both caps flat. Lines connecting RFID Integration Description Symbols with each other (for example, Switches to Readers) are normal lines and also have both caps flat.

The graphical picture consisting of Process Description Symbols together with RFID Integration Description Symbols detailing the underlying RFID infrastructure is called a *Full View* in the RFPML Studio parlance (see Figure 3.3). If we hide the RFID Integration Description Symbols in the Studio we get a *Process View* (Figure 3.6), and if we hide the Production Process Description Symbols, we correspondingly get an *RFID View* (Figure 3.7).

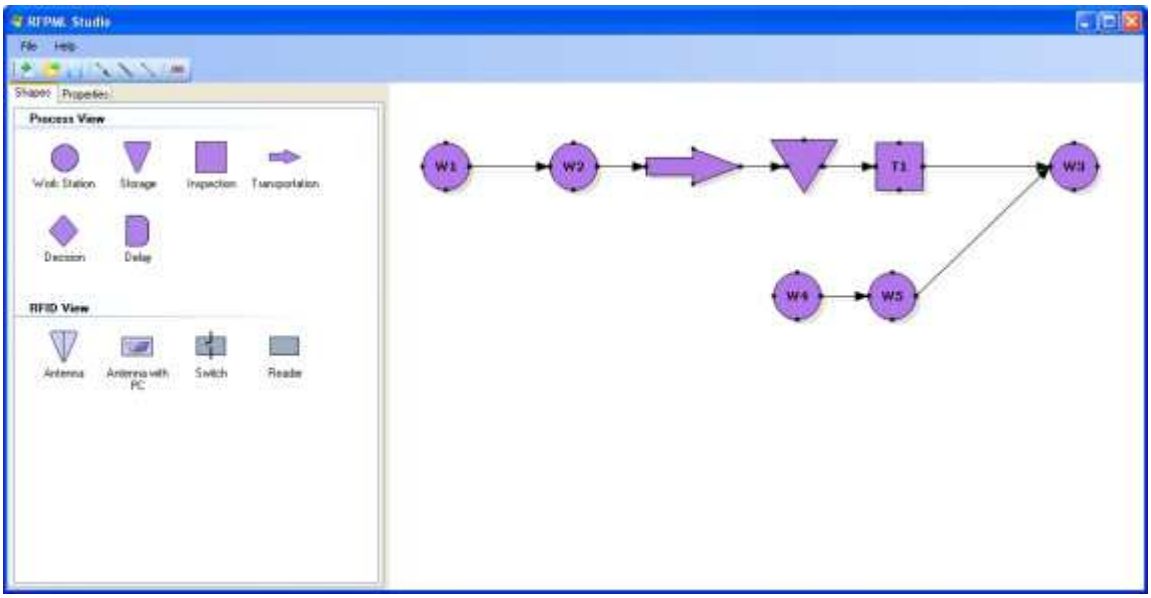

Figure 3.6: The Process View of an example production process

So far it was a description of a graphical notation of the RFPML format. There is also an equivalent textual notation which is XML-based. It is this textual language that is used for serialization and for code generation purposes. The details of this language are given in Appendix A.1.

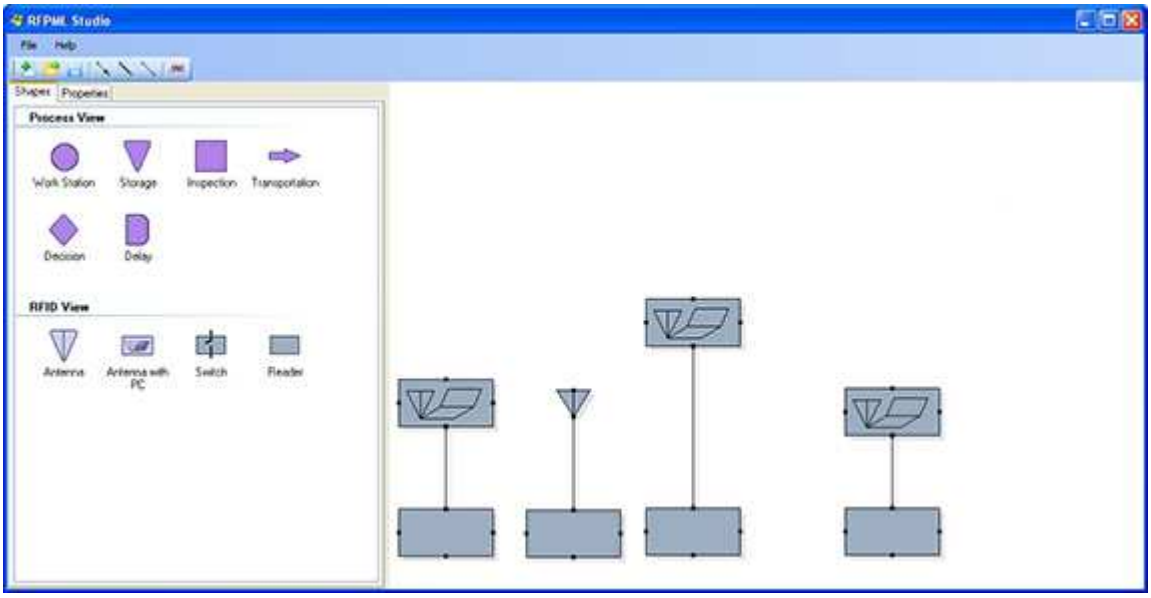

Figure 3.7: The RFID View of an example production process

3.4 Runtime

The Runtime part is the main execution engine of our system. Runtime mainly consists of 2 types of software as mentioned earlier: client and server software.

3.4.1 Client Software

Client software is the software that works on the shop floor workstation PC's. Since it works "on the edge" of the system we also refer to them as Edge Components. Edge components perform connections to the Server and provide necessary interactions with an operator if needed. The can be many client software modules, of course, as opposed to one server module.

First of all let's talk about why we need client software at all. Normally when you design a model of a process you should try to attach to each process element under consideration one and only one antenna. In this way there would be no need for client software as the information would be collected and sent to the server automatically. But there are several situations where an operator (person performing a job on a workstation) support is required through his/her interaction with the client software:

- Several stations may me connected to a single Antenna with PC. Also several

workstations may be positioned in a limited space, so if we were to attach a separate antenna to each workstation the antennas might see several workstations at once. In these situations we need some means to differentiate between workstations, so we use the client software for this purpose.

- Inspection stations. Inspection stations typically have a result associated with an operation (or testing) on that station. To enter such results to the system we again need the help of an opertor, so we use the client module. To facilitate even further this operation we provide the operator with the list of possible results for that particular station. The list can be configured during the model development phase using the RFPML Studio.

- So called *Initial Tagging Stations*. Initial Tagging Stations are the points in the production process flow where we start to collect information. Tags are attached to the parts entering production line at these points. During this operation we also need the information about the relevant job order for which the product (corresponding to the tagged part) is being manufactured. The attached tag is linked to the job order number and that information is sent to the database on the server. Again to facilitate this operation we provide the operator with the list of available job orders fetched from the *JobOrders* database table.

We tried to provide a very simple graphical user interface (GUI) for client modules bearing also in mind that the modules may also be deployed on a touchscreen-enabled PC. We also provided identical versions for different client platforms and operating systems, such as Windows XP, Vista (Figure 3.8), Windows CE (Figure 3.9). So the client software may be deployed on normal desktop PC's, notebook computers and portable (handheld) devices, such as Pocket PC, PDAs, etc.

3.4.2 Server Software

As opposed to the client software, server software is installed on one computer in the system. Among primary goals of the server software are communication with a

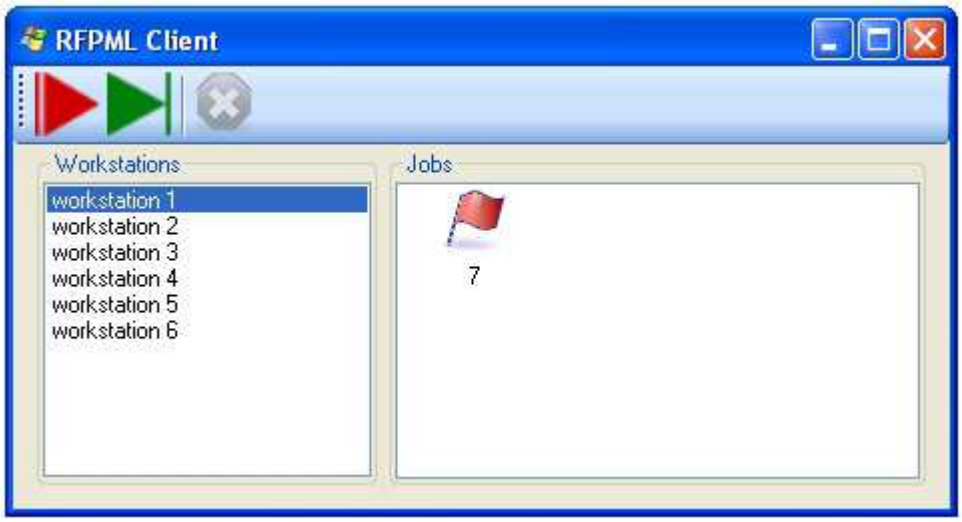

Figure 3.8: Example Windows Forms .NET client

client, control of RFID readers, processing and synchronization of data coming from readers and clients, database-related operations, etc.

When processing data coming from clients the server compares the data with the information from the database and checks for errors. Actually one of the features of our system is to control the material flow inside a production process. After the initial tagging phase the sequence of workstations through which the tagged part will pass is known from the RFPML model. In other words, we would like to guarantee that parts travel in the factory as they are supposed to. So when the part passes a workstation out of order a message warns the operator. There are many more checks as well, like, e. g., when an unidentified tag comes to a non-initial tagging station. In Section 3.4.6 an algorithm is given describing server actions in response to client messages.

In the above statement we mentioned that the information about the sequence of workstations through which a tagged part passes is known from the RFPML model. Actually it is *generated* from the RFPML model and afterwards it is saved in the database in the special column named *OrderSequence* of the table named *JobOrders*. The format of the column is the following:

$$w1, w2, w3, w10; w5, w6, w10; w7, w8, w9, w10 \tag{3.1}$$

Figure 3.9: Example Windows CE.NET client

where w*Number* denotes a workstation, comma separates workstations within a sequence and semicolon separates sequences. So in the above example there are 3 sequences *(w1,w2,w3,w10)*, *(w5,w6,w10)* and *(w7,w8,w9,w10)*. We can reconstruct the hierarchical structure of workstations for the above example as the following:

As it is seen from Figure 3.10 an example product has three parts. The first one goes through the stations w1, w2, w3. The second part goes through stations w5, w6. The third part goes through w7, w8, w9. The three parts get assembled at station w10.

Figure 3.10: Hierarchical structure of workstations for the example sequence of workstations

3.4.3 Database Structure

As mentioned earlier our system uses a database as a persistence media. All the necessary database components such as tables, rows, etc. are generated after the modeling phase using a special generating module of our graphical environment. The module uses a produced RFPML document. After the generation phase the user is ready to deploy the generated software solution in an actual manufacturing environment.

This structure of the database is shown in Figure 3.11. As it is seen from the figure it is quite simple and consists of only 3 tables:

- *JobOrders*. This table is needed for keeping information about job orders. Only two fields of the table are directly related to our system: *OrderID* - for distinguishing between different job orders; and *OrderSequence* - for keeping the sequence of workstations through which a tagged part will pass as described earlier in Section 3.4.2. Other fields are for convenience only and may be used in case there are no other (more sophisticated) order management systems available.

- *Events*. This table is used to keep track of times when a tagged part passes a workstation. *OrderId* field denotes a job order to which a given tagged part belongs, *TagCode* denotes the tag data (ID) of the part, *Source* denotes a workstation passed, *StartTime* and *FinishTime* fields are related to the concept we briefly described at the beginning of this Chapter in Section 3.1: we told there that concerning the data collection pattern there are 2 types of workstations:

workstations collecting only departure times and those collection both the arrival and departure times. *StartTime* and *FinishTime* fields store those two times. In case when we want to store only departure time, *StartTime* field is set to NULL value.

- *EventDetails*. In Section 3.4.1 we described stations which can have an associated result after processing. This table is used to record extra event information coming from this type of stations. So *EventId* just points to an event from the *Events* table, and *EventDetails* field stores the extra information related to this event.

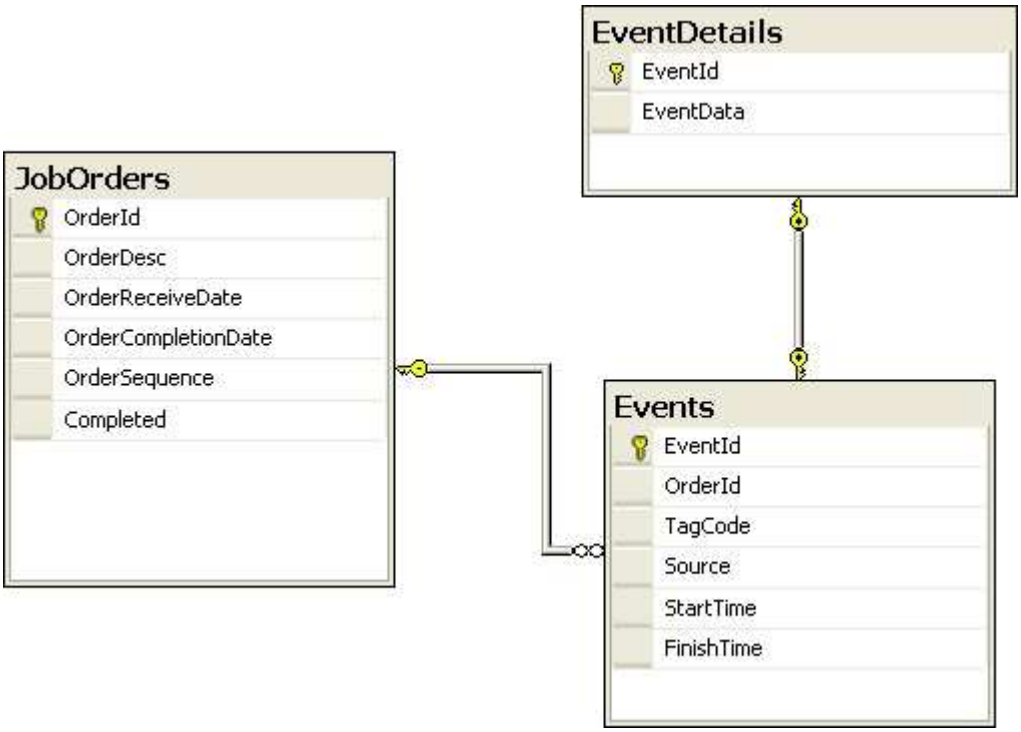

Figure 3.11: Database Structure

3.4.4 Client and Server configuration files

In this subsection we would like to talk about client and server configuration files. Client configuration files are generated by the RFPML Studio (Figure 3.12), and the server configuration file is generated by a special configuration and management

tool (Figure 3.13), which resides on the server machine. Examples of both files are given in Appendices B.1 and B.2 respectively.

Client configuration file is XML-based and has the following sections:

- *Server*. This section contains TCP information about the server machine, such as server's IP and port number.

- *Reader*. This section contains information about the reader through which data is collected. Attributes for this element include only the name of the reader.

- *Workstations*. This is a collection of *Workstation* elements, each of which contains information about all stations serviced by the respective client software module. The information consists of the name of the workstation, boolean value indicating whether it is an initial tagging station (to be described in the next subsection) or not and again a boolean value showing whether the events on the workstation have both arrival times and departure times or only departure times as described in Section 3.4.3.

Server configuration file is also an XML-based file, containing information about the RFID readers controlled by the server module. It is basically a collection of *Reader* elements having the following components:

- *Name*. The name of the reader. This name must be the same as the one in the respective client configuration files.

- *IPAddress*. IP address of the reader.

- *Port*. The port through which server connects to the reader.

- *Login*. The login name with which server connects to the reader.

- *Password*. The password with which server connects to the reader.

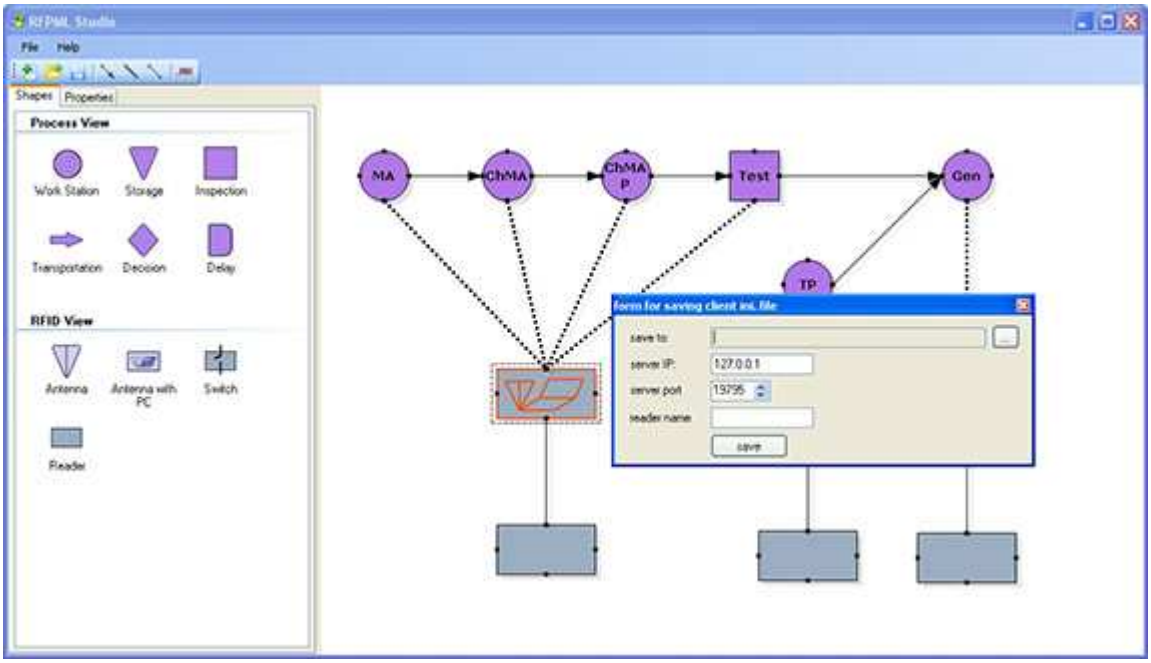

Figure 3.12: A sample client configuration file saving dialog

3.4.5 Client-Server Interaction Protocol

Before describing the Client-Server Interaction Protocol and Algorithm we would like to summarize what was said in the previous sections and to introduce some concepts which would help us in describing the protocol and the algorithm.

In Section 3.4.1 we talked about the necessity of client software and already described the types of workstations within the client-server interaction scope. We have to make a couple of points clearer here. Firstly, we differentiate between the client workstation types according to the information they send to the server. So we have only 3 types:

- *Initial Tagging Station* (or *First Tagging Station*). It is the first workstation in any sequence in a given job order, e. g. the first tagging stations for Figure 3.10 are *w1*, *w5* and *w7* for the sequences (*w1*, *w2*, *w3*, *w10*), (*w5*, *w6*, *w10*) and (*w7*, *w8*, *w9*, *w10*). Operator applies a tag to a part and connects the tag with a corresponding job order by choosing a job order ID from a list supplied to the client by the server. The information sent to the server is a job order ID and, possibly, tag data (we will explain this point a bit later).

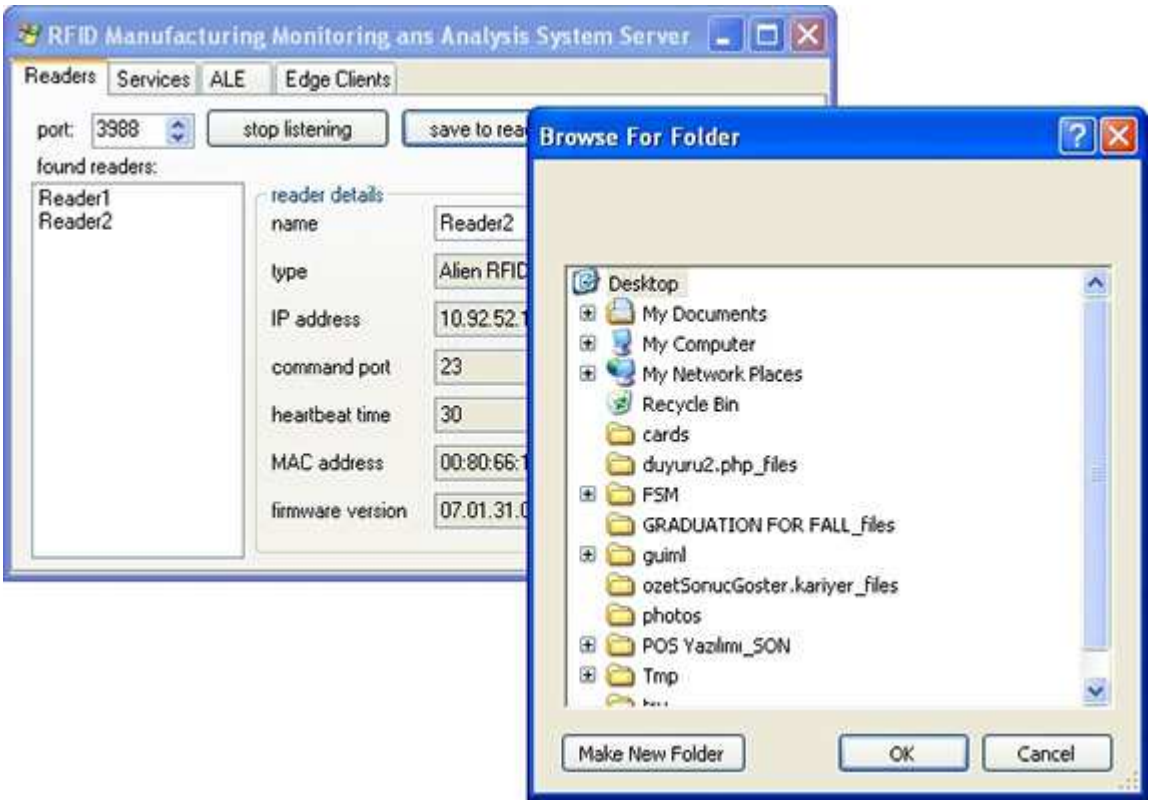

Figure 3.13: A sample server configuration file saving dialog

- *A station with an output result.* Some stations can have additional information after processing. All possible outcomes for specific stations are taken from client initialization files which are generated from the model developed in the RFPML Studio. So operator just chooses the appropriate result and sends it to the server.

- *A regular, or normal station.* Operator just signals that the part is at a specific workstation, that's all, nothing is sent to the server except, possibly, tag data (will be explained shortly). There is actually one more type called "a merging station", i. e. a station where at least 2 parts are being assembled. But since we do not send anything to the server, except, possibly, tag data, this type is actually a normal station.

We several times mentioned now that we possibly send tag data to the server. This is related to two modes of acquiring tag data:

- Tag reading is performed by the server.

- Tag reading is performed directly by the client and then the tags read are sent to the server in the body of the message.

Different clients can have different modes of tag acquisition. The modes are specified during RFPML model development phase and are recorded in the generated client initialization files.

Now we can describe the protocol itself. It is a special XML-based protocol for the interaction between clients and the server. Server acts as a TCP listener and clients occasionally connect to it to exchange information. These messages are structured in XML format (Figure 3.14) whose schema is given in Appendix A.2. The main parts of the message are as following:

```
<?xml version="1.0" encoding="utf-8"?>
<MamasMessage xmlns:xsi="http://www.w3.org/2001/XMLSche
xmlns="http://mamas.sabanciuniv.edu/MamasRequest.xsd">
  <MessageType>
    NormalPass
  </MessageType>
  <MessageData>
    <OperationType>Single</OperationType>
    <Workstation></Workstation>
    <Reader>Reader1</Reader>
    <Job>3</Job>
    <OutputResult>Success</OutputResult>
    <Tags>
        <Tag>3008 33B2 DDD9 0140 3505 0007</Tag>
    </Tags>
  </MessageData>
</MamasMessage>
```

Figure 3.14: Example of an XML-based client request file

MessageType. As the name suggests it describes the type of a message. It is of enumeration type and can have 1 of 4 values:

GetJobList. Used when a client requests a list of available job orders from the server.

InitialTagging. Used when client sends data for an initial tagging workstation to the server.

OuputPass. Used when client needs to attach some additional information when sending data to the server.

NormalPass. Used for all other messages.

MessageData. Contains data for a message. The data varies for different message types. For *GetJobList* type of messages we have the following subparts:

JobType. The more appropriate name for this would be *WorkstationType* since it designates the type of workstation requesting job order list from the server. It is of enumeration type having only 2 values: *FTS* - for initial tagging stations and *Other* - for all others.

Workstation. It is just the name of a client workstation from which the message is sent to a server.

For all other types:

OperationType. Describes the type of operation when sending data to server. It is of enumeration type having 3 values:

Single. Operator sends message only once, typically when he completes his task.

Start. Used in a so called "start-finish" scenario when operator sends 2 messages designating the start and end of his task. So this operation type is used for messages before the operator begins the task.

Finish. This operation type is used for messages after the operator finishes his task.

Workstation. Like for the *GetJobList* type of messages it is just the name of a client workstation from which the message is sent to a server.

Reader. It is the name of the reader used to read tags.

Job. It is the job order ID for which the current request occurs.

OutputResult. Some workstations can have an associated result from the work being performed on them (like, for example, error message when an error occurs). This field is used to convey this result to the server.

Tags. This field is used when tags are read by the clients themselves and is used to send read tags to the server.

3.4.6 Client-Server Interaction Algorithm

We devised a special algorithm for the Client-Server Interaction Protocol described above (see figure below). We would like to make some comments for the figure regarding figure symbols:

* Red cross represents exception to be thrown by the runtime.
* Green/red lines denote boolean responses to the previous inquiry (YES/NO respectively).
* If inquiry is not of boolean type possible responses are explicitly given in dashed rectangles.
* Green rectangles denote successful terminations of requests.

Describing the algorithm in plain words:

(1) After a request is received from the client by means of a TCP channel it is parsed to reveal any possible errors in the format.

(2) Then we check the type of the request.

  (A) if it is of *GetJobList* type return the list of available job orders. By *available* we mean either non-started jobs or jobs having non-started sequences.

  (B) Otherwise read tags if it is configured to be read by the server. Check the number of tags read.

    (I) If no tags read throw an exception

    (II) If 1 tag is read then again check the type of the request.

      (a) If it is of the *FTS* type (First, or Initial Tagging Station), check whether the tag under consideration is active or not. By *active* we mean that the tag is being used in an unfinished job order.

        (i) If it is active throw an exception.

        (ii) If it is not active register the event.

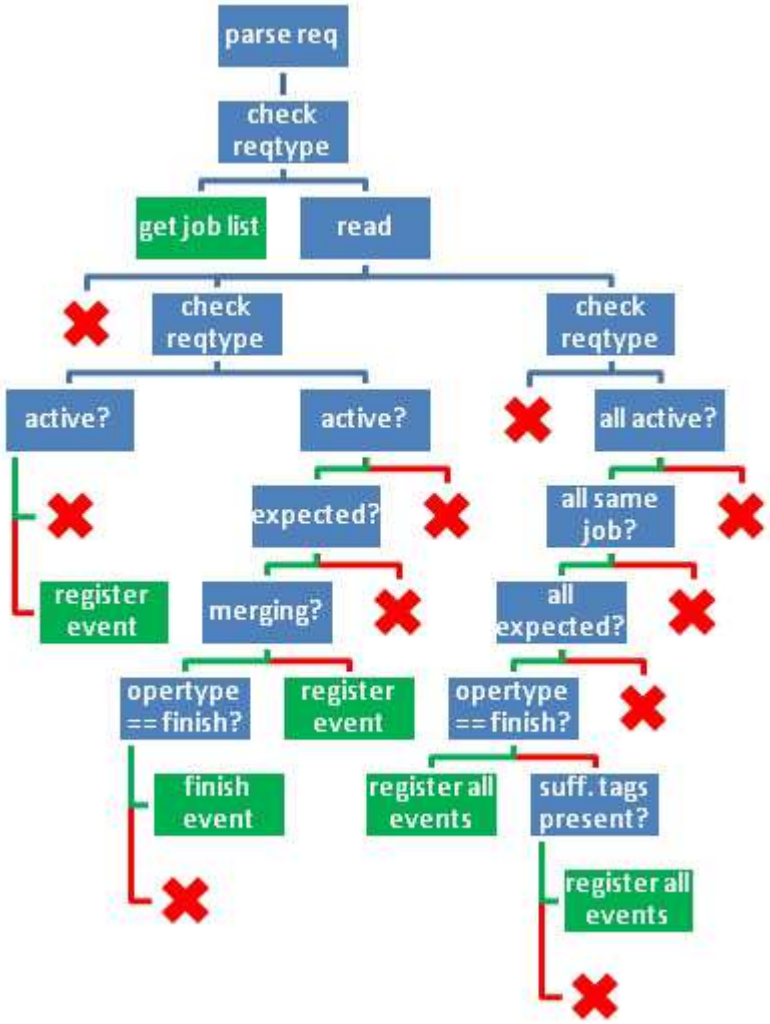

Figure 3.15: Client-Server Interaction Algorithm

(c) If it is not of the *FTS* type check whether the tag is active. Then check whether it is expected. By *expected* we mean that it comes in the correct order in the sequence so that all previous workstations were visited. Then check whether it is at the merging point according to the model. If any of the checks fails throw an exception.

(i) If it is at the merging point check whether the operation type is *Finish*.

* If the operation type is *Finish* it means that we should finish the merging event started before.

* If the operation type is not *Finish* throw an exception.

(ii) If it is not at the merging point register an event.

(III) If more than 1 tag is read it means that it should be a merging point where several parts are assembled into one. Again check the type of the request.

(a) If it is not of the *NormalPass* type throw an exception.

(b) If it is of the *NormalPass* type check whether all tags are active. Then check whether they are all from the same job order. Then check whether they are all expected. If any of the checks fails throw an exception. Then check whether the operation type is *Finish* for all tags.

(i) If the operation type is *Finish* for all tags register all events.

(ii) If the operation type is not *Finish* for all tags check whether there are sufficient tags present.

* If sufficient tags are present register all events.

* Otherwise throw an exception.

3.4.7 RFPML Web Monitoring Tool

As mentioned above a web-based monitoring tool was developed for the system. It uses an RFPML document to draw the underlying system in the browser window. A PHP library called GD [39] was used to draw all the graphics. The information about each object is shown in the pop-up box (*Information*) when you click on the shape. This pop-up box is updated with a predefined refresh rate through using AJAX technology.

3.5 Implementation

3.5.1 RFPML Studio

We built RFPML Studio on top of Microsoft .NET 2.0 Framework. For diagramming purposes we used **Netron** - free .NET diagramming library written in C#. The

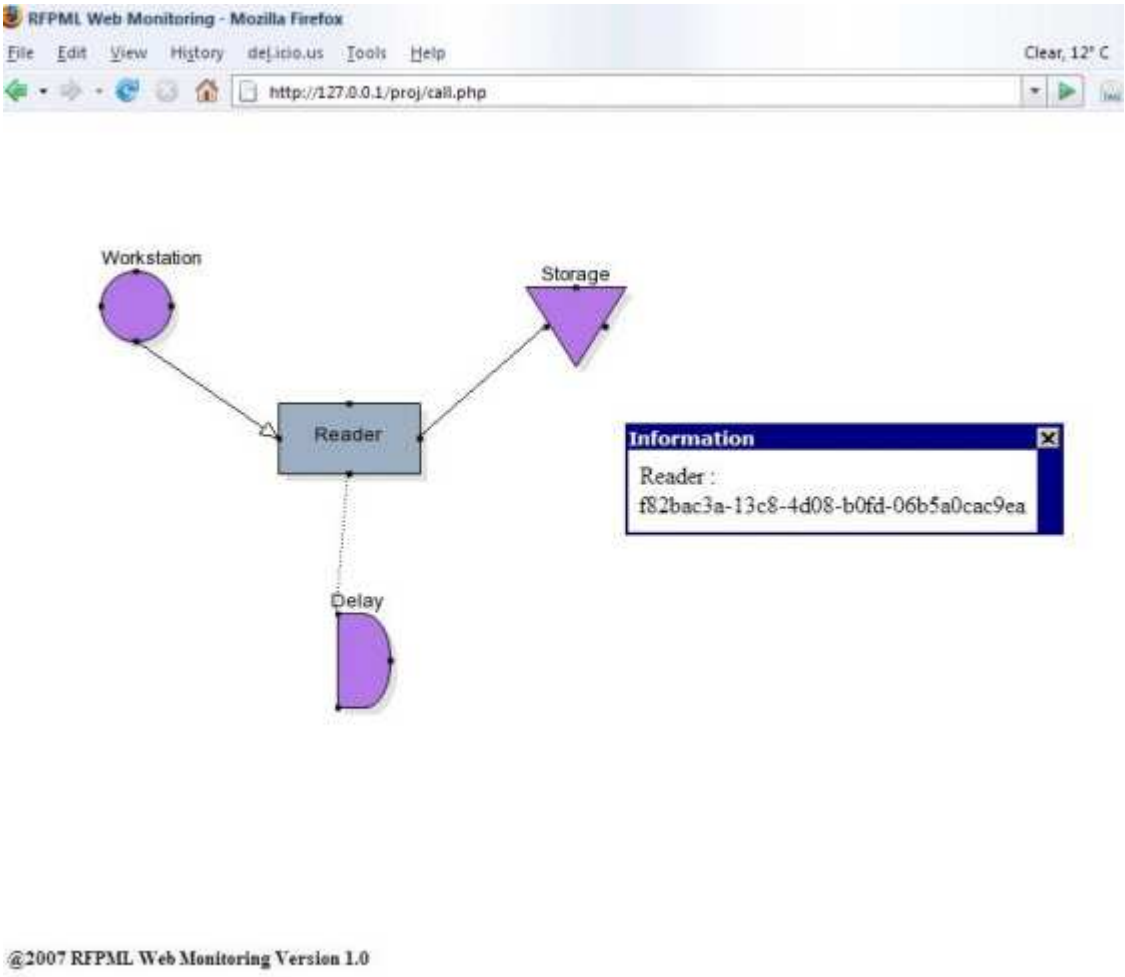

Figure 3.16: RFPML Web Monitoring Tool

biggest change we had to introduce was development of several shape types, one for each symbol type in our system, development of 3 connection types and also development of custom logic, such as when, for instance, InterConnection cannot join two objects both from the Process View or both from the RFID Infrastructure View, one of them must be from the Process View, another from the RFID View.

Regarding the class structure. There are 3 projects inside the solution:

* **Netron.Core**. The diagramming base from the Netron project mentioned above.

* **PropertyGridEx**. This component is needed to provide support for property grid used when editing properties for diagram components.

* **RfpdlStudio**. This is the main project for the RFPML Studio. The class structure of the project is the following (we omit system helper classes):
  * *MainForm*. Class containing the main form of the application.
  * *Serializer*. Class for serializing models into .rfpml format.
  * *IniGenerator*. Class for generating initialization files for clients.
  * Several helper classes for serialization.

3.5.2 Runtime

The runtime was also implemented on the .NET Framework 2.0 platform. The main solution, called MaMAS (which stands for **Ma**nufacturing **M**onitoring and **A**nalysis **S**ystem) consists of several projects:

* **Service**. A Windows Service which provides all the necessary servicing functionality to edge clients. It handles requests from the clients, sends back responses, performs database queries, reads tags from correct readers if necessary and enhances business logic of the underlying manufacturing process. It implements the server endpoint of the above-mentioned Client Server Interaction Protocol.
* **Client**. A WinForms.NET client which acts as a frontend for the manufacturing shop floor. It takes user input when necessary and provides interaction with the server. It implements the client endpoint of the above-mentioned Client Server Interaction Protocol.
* **CeClient**. A Windows CE.NET client with the same functionality as the above WinForms.NET client.
* **Server**. Server provides management of all related hardware and software as well as serves as the point of communication with the RFPML Studio.
* **Common**. This projects contains common constants, enumerations, classes, etc. which are shared between the Client/CeClient and Service.

Chapter 4

Factory Integration

The pilot implementation of the developed system was executed at some critical workstations of the generator assembly process at Teksan Generator Company. Teksan is one of the leading and most advanced producers of diesel generating sets. They offer more than 300 different models ranging from 0.5 kVA to 3500 kVA. In their factory at Sarıgazi, Istanbul, they seek a higher degree of visibility and control over certain stages throughout the assembly of the main components and sub-assemblies of the generators. It is also worth noting that the production environment at Teksan is a very challenging one to model and deploy an RFID-based system since the facility is dense with metallic materials.

The production of generator is a fairly complex process. The complexity is even increased because of the customer-based model of production adopted at Teksan. There are many customization points depending on customer's choices. A simplified diagram of a generator production flow is shown in Figure 4.1.

With the help of Teksan's management we identified the stations (denoted by a red square on the diagram) of particular interest since at those stations errors were frequently encountered, like, for example, when parts from different job orders were combined to form a new system.

Abbreviations of station names mean the following:

MA: a station where a coupling of an engine and an alternator occurs. It is an Initial Tagging Station.

ChMA: a station where assembly of MA group on the chassis occurs.

ChMAP: a station where assembly of ChMA group and control panel occurs.

Test: a station where the assembled system passes through tests.

TP: a station where a transfer panel is assembled. In our model it is again an Initial Tagging Station.

Gen: a station where a transfer panel is attached to the ChMAP group and a generator assembly is finished.

The corresponding model (Figure 4.2) was constructed using our RFPML Studio. We depicted only the stations which were of interest to the factory staff. All stations are naturally of the *Workstation* type, except for the *Test* station, which is an *Inspection* station.

We have only two sequences of stations to monitor, that is (*MA*, *ChMA*, *ChMAP*, *Test*, *Gen*) and (*TP*, *Gen*). Initial tagging stations are *MA* and *TP* respectively. At *Gen* the two sequences join and appropriate control is executed at runtime.

We decided, as it is also can be seen from Antennas with PC of the model, to have 3 client software modules, since we had only three RFID readers available, two Alien 8800 model fixed readers and one CAEN A528 integrated Psion Teklogix Workabout Pro handheld device. Information from stations *MA*, *ChMA*, *ChMAP* and *Test* was collected using the handheld reader, from other stations - using the fixed readers. This choice was made because these 4 stations were physically located relatively close to each other, while TP was even on the other floor.

We would also like to mention the time it took us to develop a whole system with the help of our system. We had all the readers plugged and devices connected to each other through a network. Construction of a diagram took approx. 10 minutes. Discovery of RFID readers through a MaMAS Configuration tool took some 2 minutes more. Deploying configuration files to client and server machines including a handheld RFID reader took another 5 minutes. Adding necessary job sequence records to a database took 3 minutes. So the whole time we spent at development and deployment of the system was not more than 25 minutes.

After three runs of generator assembly process the data shown in Figure 4.3 were obtained.

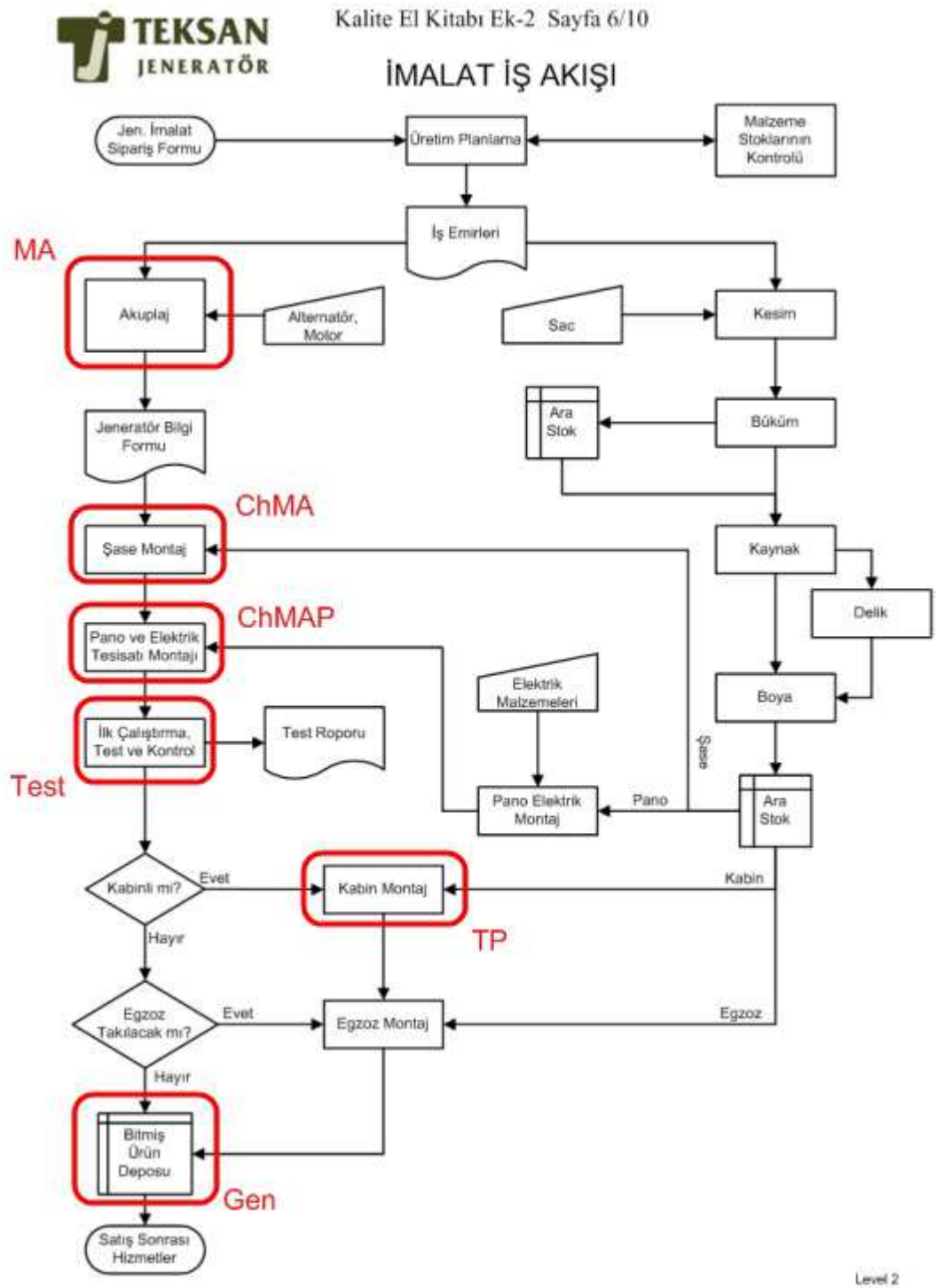

Figure 4.1: Production flow for a generator. Courtesy of Teksan Generator Factory.

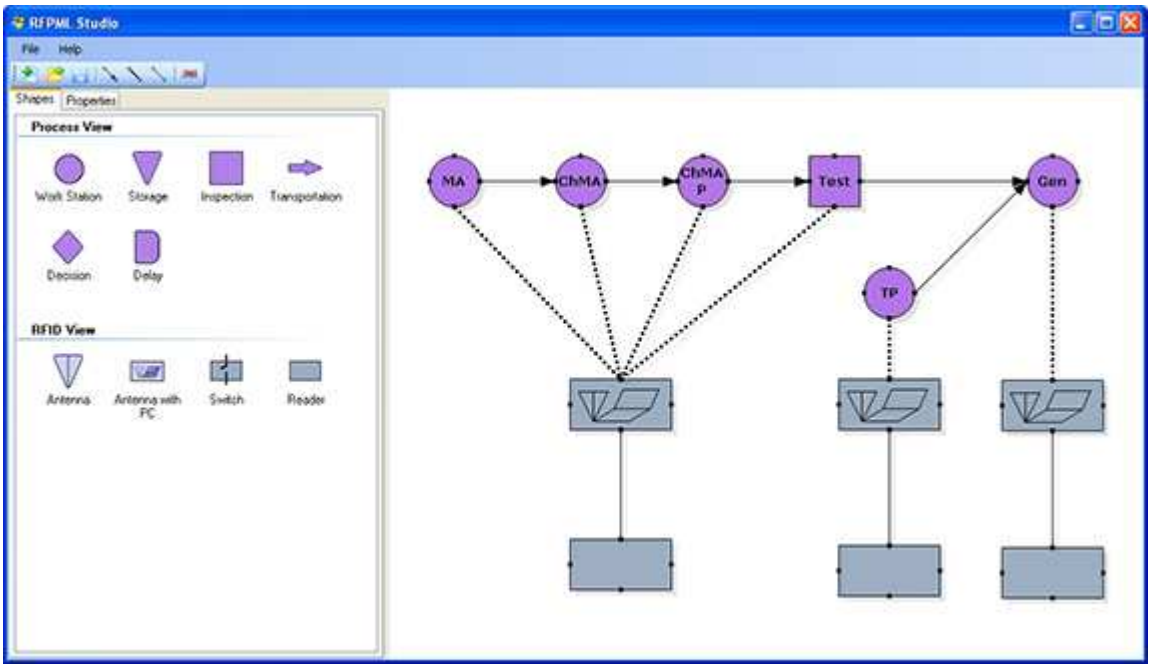

Figure 4.2: A Full View of the generator production process of the Teksan factory

EventId	OrderId	TagCode	Source	StartTime	FinishTime
350	8	1978 33B2 DDD9 0140 3505 1979	MA	27.02.2008 10:09:45	27.02.2008 10:47:54
351	8	1978 33B2 DDD9 0140 3505 1979	ChMA	27.02.2008 11:16:07	27.02.2008 11:16:07
352	8	1978 33B2 DDD9 0140 3505 1979	ChMAP	27.02.2008 13:16:15	27.02.2008 13:16:15
353	8	1978 33B2 DDD9 0140 3505 1979	Test	27.02.2008 13:23:49	27.02.2008 14:31:26
354	8	3008 33B2 DDD9 0140 3505 0000	TP	27.02.2008 12:21:21	27.02.2008 12:21:21
355	8	3008 33B2 DDD9 0140 3505 0000	Gen	27.02.2008 14:35:37	27.02.2008 14:35:37
356	8	1978 33B2 DDD9 0140 3505 1979	Gen	27.02.2008 14:35:37	27.02.2008 14:35:37
357	9	1978 33B2 DDD9 0140 3505 1980	MA	27.02.2008 10:33:45	27.02.2008 11:11:07
358	9	1978 33B2 DDD9 0140 3505 1980	ChMA	27.02.2008 12:39:24	27.02.2008 12:39:24
359	9	1978 33B2 DDD9 0140 3505 1980	ChMAP	27.02.2008 14:44:36	27.02.2008 14:44:36
360	9	1978 33B2 DDD9 0140 3505 1980	Test	27.02.2008 14:48:49	27.02.2008 15:33:26
361	9	3008 33B2 DDD9 0140 3505 0001	TP	27.02.2008 12:29:42	27.02.2008 12:29:42
362	9	3008 33B2 DDD9 0140 3505 0001	Gen	27.02.2008 15:37:13	27.02.2008 15:37:13
363	9	1978 33B2 DDD9 0140 3505 1980	Gen	27.02.2008 15:37:13	27.02.2008 15:37:13
364	10	1978 33B2 DDD9 0140 3505 1981	MA	27.02.2008 11:21:45	27.02.2008 11:59:07
365	10	1978 33B2 DDD9 0140 3505 1981	ChMA	27.02.2008 12:27:24	27.02.2008 12:27:24
366	10	1978 33B2 DDD9 0140 3505 1981	ChMAP	27.02.2008 14:59:36	27.02.2008 14:59:36
367	10	1978 33B2 DDD9 0140 3505 1981	Test	27.02.2008 15:04:49	27.02.2008 16:04:26
368	10	3008 33B2 DDD9 0140 3505 0002	TP	27.02.2008 13:35:51	27.02.2008 13:35:51
369	10	3008 33B2 DDD9 0140 3505 0002	Gen	27.02.2008 16:08:48	27.02.2008 16:08:48
370	10	1978 33B2 DDD9 0140 3505 1981	Gen	27.02.2008 16:08:48	27.02.2008 16:08:48

Figure 4.3: Tag event data table obtained for 3 generator assembly process runs

Chapter 5

Conclusions and Future Work

In this thesis we present a generic manufacturing monitoring and analysis system allowing a rapid development of real-time manufacturing automation solution through the use of RFID technology. We have implemented the graphical modeling environment of the proposed system, which is used to generate the necessary software components ready to be deployed on the computers at the shop floor and a central server. It took approx. 25 minutes for us to develop and deploy a test system on a shop floor in Teksan Generator factory.

As a future research on this topic we plan to add some analytic capabilities to the system and develop a generic API to enable fuller integration with external information systems such as ERP or MES.

One of the possible extensions for the RFPML Studio would be the ability to include custom user-defined libraries of both Process and RFID Integration Description symbols. At present we have only a predefined set, which can not be changed. Another improvement for the RFPML Studio would be development of a more solid generic modeling framework such as, for example, the Generic Modeling Environment [40] or the Ptolemy Project [31].

Appendix A

Descriptions and XML Schemas of some of the formats developed for the system

A.1 RFPML Document Format

`<rfpml>`: the main element of an RFPML document. Used in order to conform to the general format of an XML document. Contains exactly one `<process>`, exactly one `<rfid>`, at most one (one or no such element at all) `<connections>` and at most one `<connectors>` elements.

`<process>`: contains the production process elements corresponding to Process Description Symbols of the graphical representation. Contains at least one `<workstation>` element, and any number (zero or more) of `<storage>`, `<inspection>`, `<delay>` and `<transportation>` elements.

`<workstation>`, `<storage>`, `<inspection>`, `<delay>`, and `<transportation>`: elements corresponding to the Process Description Symbols of the graphical representation. They all have the following set of attributes:

- *id*. The id of an element. Must be unique throughout the document.

- *text*. The name of an element. Optional.

- *description*. The description of an element. Optional.

- *locX*. The *X* coordinate of an element on a document canvas.

- *locY*. The *Y* coordinate of an element on a document canvas.

The `<inspection>` element contains in addition at least 2 `<outcome>` elements, which denote the possible outcomes for the *Inspection* type of stations.

`<rfid>`: contains the elements corresponding to RFID Integration Description Symbols of the graphical representation. Contains at least 1 `<antenna>` and at least 1 `<reader>` elements, and any number of `<switch>` elements.

`<antenna>`, `<reader>`, and `<switch>`: elements corresponding to RFID Integration Description Symbols of the graphical representation. They all have the same set of attributes as production process elements, i. e. id, text, description, locX, and locY. `<antenna>` in addition has a `<type>` attribute, which is of enumeration type having 2 values: *normal* and *pc*.

`<connections>`: an element denoting the connections - the objects connecting various symbols, both Process Description and RFID Integration Description, with each other. Contains at least 1 `<connection>` element.

`<connection>`: an element described in the previous paragraph. Has 4 attributes:

- *id*. The id of an element. Must be unique throughout the document.

- *from*. The id of a `<connector>` element from which this `<connection>` is exiting.

- *to*. The id of a `<connector>` element to which this `<connection>` is entering.

- *type*. The type of a connection. It is of enumeration type having 3 possible values: *process*, denoting connections between Process Description Symbols, *rfid*, denoting connections between RFID Integration Description Symbols, and *inter*, denoting connections between symbols of different domains.

`<connectors>`: an element denoting the connectors - the objects used by connections to connect to Process or RFID Integration symbols or to other connections. Contains at least 1 `<connector>` element.

`<connector>`: an element described in the previous paragraph. Has 4 attributes:

- *id*. The id of an element. Must be unique throughout the document.

- *parent_id*. The id of a parent of the current `<connector>` element. If a `<connector>` has a parent, it is not drawn on a canvas. These situations happen, for example, when a connection line is broken, so it in reality consists of two connections linked to each other.

- *locX*. The X coordinate of an element on a document canvas.

- *locY*. The Y coordinate of an element on a document canvas.

The corresponding XML Schema of the format:

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema id="RFID"
        targetNamespace="http://mamas.sabanciuniv.edu/RFPML.xsd"
        elementFormDefault="qualified"
        xmlns="http://mamas.sabanciuniv.edu/RFPML.xsd"
        xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="rfpml">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="process">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="workstation" maxOccurs="unbounded">
                <xs:complexType>
                  <xs:sequence />
                  <xs:attributeGroup ref="item_attribs" />
                </xs:complexType>
              </xs:element>
              <xs:element name="storage" maxOccurs="unbounded" minOccurs="0">
                <xs:complexType>
                  <xs:sequence />
```

```
            <xs:attributeGroup ref="item_attribs" />
          </xs:complexType>
        </xs:element>
        <xs:element name="inspection" minOccurs="0"
                    maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="outcome" type="xs:string" minOccurs="2"
                          maxOccurs="unbounded" />
            </xs:sequence>
            <xs:attributeGroup ref="item_attribs" />
          </xs:complexType>
        </xs:element>
        <xs:element name="delay" minOccurs="0" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence />
            <xs:attributeGroup ref="item_attribs" />
          </xs:complexType>
        </xs:element>
        <xs:element name="transportation" minOccurs="0"
                    maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence />
            <xs:attributeGroup ref="item_attribs" />
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="rfid">
    <xs:complexType>
```

```
        <xs:sequence>
          <xs:element name="antenna" maxOccurs="unbounded">
            <xs:complexType>
              <xs:sequence />
              <xs:attributeGroup ref="item_attribs" />
              <xs:attribute name="type">
                <xs:simpleType>
                  <xs:restriction base="xs:string">
                    <xs:enumeration value="normal" />
                    <xs:enumeration value="pc" />
                  </xs:restriction>
                </xs:simpleType>
              </xs:attribute>
            </xs:complexType>
          </xs:element>
          <xs:element name="switch" maxOccurs="unbounded" minOccurs="0">
            <xs:complexType>
              <xs:sequence />
              <xs:attributeGroup ref="item_attribs" />
            </xs:complexType>
          </xs:element>
          <xs:element name="reader" maxOccurs="unbounded">
            <xs:complexType>
              <xs:sequence />
              <xs:attributeGroup ref="item_attribs" />
            </xs:complexType>
          </xs:element>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="connectors" minOccurs="0">
```

```
    <xs:complexType>
      <xs:sequence>
        <xs:element name="connector" maxOccurs="unbounded"
                    minOccurs="1">
          <xs:complexType>
            <xs:sequence>
            </xs:sequence>
            <xs:attribute name="id" type="xs:string" />
            <xs:attribute name="parent_id" type="xs:string" />
            <xs:attribute name="locX" type="xs:int">
            </xs:attribute>
            <xs:attribute name="locY" type="xs:int" />
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="connections" minOccurs="0">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="connection" maxOccurs="unbounded"
                    minOccurs="1">
          <xs:complexType>
            <xs:sequence>
            </xs:sequence>
            <xs:attribute name="id" type="xs:string" />
            <xs:attribute name="from" type="xs:string" />
            <xs:attribute name="to" type="xs:string" />
            <xs:attribute name="type">
              <xs:simpleType>
                <xs:restriction base="xs:string">
```

```
                        <xs:enumeration value="process" />
                        <xs:enumeration value="rfid" />
                        <xs:enumeration value="inter" />
                      </xs:restriction>
                    </xs:simpleType>
                  </xs:attribute>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:attributeGroup name="item_attribs">
    <xs:attribute name="id" type="xs:string" />
    <xs:attribute name="text" type="xs:string" />
    <xs:attribute name="description" type="xs:string" />
    <xs:attribute name="locX" type="xs:int" />
    <xs:attribute name="locY" type="xs:int" />
  </xs:attributeGroup>
</xs:schema>
```

A.2 Client Request Message Format for the Client-Server Interaction Protocol

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema id="Request"
        targetNamespace="http://mamas.sabanciuniv.edu/Request.xsd"
        elementFormDefault="qualified"
        xmlns="http://mamas.sabanciuniv.edu/Request.xsd"
```

```
        xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="MamasMessage">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="MessageType" type="xs:string" />
        <xs:element name="MessageData">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="OperationType" type="xs:string" />
              <xs:element name="Workstation" type="xs:string" />
              <xs:element name="Reader" type="xs:string" />
              <xs:element name="Job" type="xs:string" minOccurs="0" />
              <xs:element name="OutputResult" type="xs:string" minOccurs="0" />
              <xs:element name="Tags" minOccurs="0">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="Tag" type="xs:string" maxOccurs="unbounde
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Appendix B

Examples files for some of XML formats developed for MaMAS Runtime

B.1 Example of a Client Configuration File

```
<?xml version="1.0" encoding="utf-8"?>
<ClientInfo xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xmlns="http://mamas.sabanciuniv.edu/MamasEdge.xsd">
  <Server ip="192.168.0.1" port="19795" />
  <Reader name="Reader1" />
  <Workstations>
    <Workstation name="MA" isInitTagging="true" hasStartEnd="false" />
    <Workstation name="ChMA" isInitTagging="false" hasStartEnd="false" />
    <Workstation name="ChMAP" isInitTagging="false" hasStartEnd="false" />
    <Workstation name="Test" isInitTagging="false" hasStartEnd="false" />
  </Workstations>
</ClientInfo>
```

B.2 Example of a Server Configuration File

```
<?xml version="1.0" encoding="utf-8"?>
<Readers xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xmlns="http://mamas.sabanciuniv.edu/Readers.xsd">
  <Reader name="Teksan Reader">
    <IPAddress>192.168.0.101</IPAddress>
    <Port>23</Port>
    <Login>alien</Login>
    <Password>password</Password>
  </Reader>
</Readers>
```

Bibliography

[1] Michael McClellan, "Optimizing the Manufacturing Enterprise Using BPM," available online at https://www.bpminstitute.org/presentations/featured-presentation/article/track-chair-keynote-optimizing-the-manufacturing-enterprise-using-bpm.html (last accessed at January 2008).

[2] Lu, B.H., Bateman, R.J. and Cheng, K., "RFID enabled manufacturing: fundamentals, methodology and applications," *Int. J. Agile Systems and Management* **1**(1), pp. 73–92, 2006.

[3] T. Sobezak, *Glossary of Terms for Computer Integrated Manufacturing*, Computer and Automated Systems Association of; 1st ed edition, June 1984.

[4] James A. Rehg, Henry W. Kraebber, *Computer-Integrated Manufacturing*, Prentice-Hall, 2001.

[5] Grier C. I. Lin, Sev V. Nagalingam, *CIM Justification and Optimization*, Taylor & Francis, 2000.

[6] Wikipedia article, "Computer integrated manufacturing," available online at http://en.wikipedia.org/wiki/Computer_Integrated_Manufacturing (last accessed at January 2008).

[7] Russel Biekert, *CIM Technology*, The Goodheart-Willcox Company, Inc., 1998.

[8] RFID Journal, "The Basics of RFID Technology," available online at http://www.rfidjournal.com/article/articleview/1337/1/129/ (last accessed at January 2008).

[9] "All Weather Asset Tracking Tag, 5-ETD00433," available online at http://www.visonictech.com/elpas_tags.html (last accessed at January 2008).

[10] "SEAL ARFID Platform based sensors," available online at http://www.x3-c.com/?id=5f1520a4-273e-102a-a307-00e029508ef8&sid=171b3e44-2eb6-102a-a307-00e029508ef8 (last accessed at January 2008).

[11] Wikipedia article, "Radio-frequency identification," available online at http://en.wikipedia.org/wiki/RFID (last accessed at January 2008).

[12] Johnson D., "RFID tags improve tracking, quality on Ford line in Mexico," *Control Engineering* **49**(11), pp. 16–16, 2002.

[13] "IBM Fishkill semiconductor plant an example of agile, realtime operation," available online at http://whitepapers.zdnet.co.uk/0,1000000651,260090942p,00.htm (last accessed at January 2008).

[14] Brewer, A. and Landers, T., "Radio Frequency Identification: A Survey and Assessment of the Technology," 1997. University of Arkansas Department of Industrial Engineering Technical Report.

[15] Shouqin Zhou, Weiqing Ling and Zhongxiao Peng, "An RFID-based remote monitoring system for enterprise internal production management," *The International Journal of Advanced Manufacturing Technology* **33**(7-8), pp. 837–844, July, 2007.

[16] Koumpis K., Hanna L., Andersson M., and Johansson M., "Wireless Industrial Control and Monitoring beyond Cable Replacement," in *PROFIBUS International Conference*, 2005.

[17] Brewer A., Sloan N., and Landers T. L., "Intelligent tracking in manufacturing," *Journal of Intelligent Manufacturing* **10**(3-4), pp. 245–250, September 1999.

[18] Huang, G. Q., Zhang Y.F., Jiang P.Y., "RFID-based wireless manufacturing for real-time management of job shop WIP inventories," *The International Journal of Advanced Manufacturing Technology* , 2005.

[19] Huang, G. Q., Zhang Y.F., Jiang P.Y., "RFID-based wireless manufacturing for walking-worker assembly islands with fixed-position layouts," *Robotics and Computer-Aided Manufacturing* **23**(4), pp. 469–477, August 2007.

[20] Qiu, R. G., "RFID-enabled automation in support of factory integration," *Robotics and Computer-Aided Manufacturing* **23**(6), pp. 677–683, December 2007.

[21] Yagi J., Arai E., Arai T., "Parts and packets unification radio frequency identification application for construction," *Automation in Construction* **14**(4), pp. 477–490, August 2005.

[22] Li Zhekun, Rajit Gadh, & B.S. Prabhu., "Applications of RFID technology and smart parts in manufacturing," in *Proceedings of ASME 2004 Design Engineering Technical Conferences and Computers and Information in Engineering Conference (DETC2004)*, 2004.

[23] Baudin M., Rao A., "RFID applications in manufacturing," available online at http://www.mmt-inst.com/RFID%20applications%20in%20manufacturing%20_Draft%207_.pdf (last accessed at March 2007).

[24] Wikipedia article, "Software agent," available online at http://en.wikipedia.org/wiki/Software_agent (last accessed at January 2008).

[25] Tnazefti-Kerkeni, I.; Arantes, L.; Paviot-Adet, E., "An agent-oriented framework for controlling and monitoring manufacturing system," in *2003 IEEE International Symposium on Intelligent Control.*, pp. 383–388, 2003.

[26] Sauer, O.; Sutschet, G., "A step towards real time [production monitoring and control system]," *Manufacturing Engineer* **85**(3), pp. 32–37, June-July 2006.

[27] Paulo Leitao, Francisco Restivo, "Agent-Based Holonic Production Control," in *13th International Workshop on Database and Expert Systems Applications (DEXA'02)*, p. p. 589, 2002.

[28] "Concepts for Holonic Manufacturing," available online at http://www.mech.kuleuven.be/goa/concepts.htm (last accessed at January 2008).

[29] Yurtsever T., Pierce N.G., "Computerized Manufacturing Monitoring and Dispatch System," *Computers and Industrial Engineering* **35**(1), pp. 137–140, October 1998.

[30] "RFID Integrated Solution Enablement," available online at http://www.alphaworks.ibm.com/tech/rise (last accessed at March 2007).

[31] J. Davis et al., "Overview of the Ptolemy Project," available online at http://ptolemy.eecs.berkeley.edu/publications/papers/01/overview/overview.pdf, Mar. 2001 (last accessed at March 2007).

[32] Simsek, O., Caglar, R. C. and Erdem, M. A., "Implementing a Visual Editor and a Monitoring Software," 2007. Sabanci University ENS 491/492 Progress Report.

[33] Mark Treat, "What is BPM Anyway?," available online at https://www.bpminstitute.org/articles/article/article/what-is-bpm-anyway.html (last accessed at January 2008).

[34] "XPDL," available online at http://www.wfmc.org/standards/xpdl.htm (last accessed at January 2008).

[35] "Process Specification Language," available online at http://www.mel.nist.gov/psl/ (last accessed at March 2007).

[36] "SensorML," available online at http://vast.uah.edu/SensorML/ (last accessed at March 2007).

[37] Sule, D. R., *Manufacturing facilities: location, planning and design*, Boston: PWS Pub. Co., 1994.

[38] Ben B. Graham, President, The Ben Graham Corporation, "Rediscover Work Simplification," available online at

http://www.worksimp.com/articles/rediscover%20work%20simplification.htm (last accessed at January 2008).

[39] "GD Library," available online at http://www.libgd.org (last accessed at January 2008).

[40] A. Ledeczi, M. Maroti, A. Bakay, G. Karsai, J. Garrett, Ch. Thomason, G. Nordstrom, J. Sprinkle and P. Volgyesi, "The Generic Modeling Environment. In Proceedings of Workshop on Intelligent Signal Processing," May 2001. available online at http://www.isis.vanderbilt.edu/Projects/gme/GME2000Overview.pdf (last accessed at March 2007).