NORWEGIAN UNIVERSITY OF SCIENCE AND TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY, MATHEMATICS
AND ELECTRICAL ENGINEERING

Implementation of the Intelligent Printing Management

Project Assignment

Maxim Langebrekke

Autumn 2004

NORWEGIAN UNIVERSITY OF SCIENCE AND TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY, MATHEMATICS AND ELECTRICAL ENGINEERING

PROJECT ASSIGNMENT

Student's name:	Maxim Langebrekke
Course:	TTM4700 Teleservices and Networks, Specialization
Title:	**Implementation of the Intelligent Printing Management**

Text:

Intelligent Printing Management (IPM) is a concept related to managing network shared printers. The basic assumption is a set of printers serving tens or hundreds of users. Such a system demands a rather high degree of configuration flexibility and must comprise mechanisms such as agreed-on-policies and priority matching. The task is to specify, design, implement and evaluate IPM using the TAPAS architecture. IPM must interface the printer service interfaces, print job queues and printing clients.

The project will comprise the handling of different printer interfaces (HP, Xerox, etc.) with different range of capabilities and features (BW/colour, resolution, duplex support, dpi, etc.).

Deadline:	November 29, 2004
Handed in:	
Carried out at:	**Department of Telematics**
Supervisor:	Mazen Malek Shiaa

Trondheim, ...2004

Finn Arve Aagensen
Professor

Preface

This report documents the results of a project assignment, “Implementation of the Intelligent Printing Management”, performed autumn 2004 by Maxim Langebrekke. The project is a part of the MSc studies at the Norwegian University of Science and Technology, Faculty of Information Technology, Mathematics and Electrical Engineering, Department of Telematics from September to December 2004.

The work with this project has been challenging, partly because I did not have much knowledge of TAPAS and of semantic web languages in general. I have, despite of the complexity of TAPAS, gained much experience and knowledge from this work that I will have advantage of for future work.

I would like to thank my supervisor Mazen Malek Shiaa, and professor Finn Arve Aagensen for their advice and support throughout the project. I would also like to thank my fellow students at the TAPAS lab for encouraging me during the difficult moments all this semester.

Trondheim, 2004-11-29

Maxim Langebrekke

Table of Contents

Figure List

Table List

Abbreviation

AFP	Advanced Function Presentation
ASCII	American Standard Code for Information Interchange
CIM	Common Information Model
CM	Configuration Manager
CSEMon	Capability, Status and Event Monitor
CSRep	Capability & Status Repository
DPI	Dots Per Inch
FIFO	First In First Out
IPP	Internet Printing Protocol
PaP	Plug-and-play
PCL	Printer Control Language
PDF	Portable Document Format
PDL	Page-Description Language
PlayRep	Play Repository
PMS	Printing Management System
PPM	Page Per Minute
RDF	Resource Description Framework
RIP	Raster Image Processor
SI	Service Installer
SR	Service Reconfigurator
UML	Unified Modeling Language
XDD	XML Declarative Description
XML	eXtensible Markup Language

Abstract

This project uses the concepts of TAPAS (Telematics Architecture for Play-based Adaptable System) to model an Intelligent Printing Management (IPM) system as an application of the developed framework and with the use of plug-and-play. The architecture framework proposed in TAPAS handles several aspects of adaptability. However, this project will focus mainly on the aspects of dynamic configuration. This project aims to specify, design, implement and evaluate IPM using the TAPAS architecture.

IPM is a concept in managing network and Internet shared printers. Network printing systems demand a rather high degree of configuration flexibility and must comprise mechanisms such as agreed-on-policies and priority matching. IPM is proposed as a solution to common printing problems and to the inefficiency in terms of printer usage and achieved tasks in printer settings. The proposed solution is a print server role with adequate intelligence. The print server role acts as a configuration manager by controlling and managing the system based on predefined configuration and reconfiguration rules.

Different aspects of TAPAS, together with other related topics have been studied before being able to describe, model and specify the IPM system. The focus has been on the required functionality needed for this system to handle the different dynamic configuration aspects. Furthermore, different data models have been proposed to describe capability and status descriptions, role specifications and descriptions of configuration and reconfiguration rules. Several system solutions have been appraised in order to understand how the system in the most efficient and quick manner can handle and manage the different print jobs. There were certain rules already defined for most of the actors and service components. These rules were basic and had to be followed in order to properly describe and model the complete system. New print server roles have been proposed to handle different new types of print jobs. As for the print jobs, it was important to always monitor them, so that they could be redirected to another printer in case of any congestion at a printer, or if any type of operational error should occur. This would prevent the print jobs from being lost.

1 Introduction

This section will give an introduction to the project, where its background, approach, goals and demarcations will be defined. This section ends by describing the project's structure.

1.1 Background

Today's telecommunication systems are increasing in complexity where installation and maintenance of these systems represents a big challenge even for qualified IT personnel. New services is being developed rapidly, and many research and development groups have realized the need to develop systems that are able to configure themselves in different environments and that support dynamic introduction of new and distributed services. These dynamic systems are also referred to the term adaptable networking: "Adaptable networking means that the provided network based services are capable of handling dynamic changes in both time and position related to resources, users and changed service requirements" [19].

TAPAS (Telematics Architecture for Play-based Adaptable System) [16] is a research project which aims is to develop an architecture concept for Plug-and-Play telecommunication equipment and services. The goal is to enhance the flexibility, efficiency and simplicity of system installation, deployment, operation, management and maintenance by enabling dynamic configuration of network components and network-based service functionality. The Department of Telematics at the Norwegian University of Science and Technology, in cooperation with SINTEF, has implemented a prototype of this architecture using the Java programming language. A lightweight version of TAPAS, called MicroTAPAS, has been specified and developed, which aims for the employment on small handheld wireless devices. The TAPAS Platform is using XML as a common representation language and the project has been running since 1997.

Another objective of the TAPAS project is to demonstrate the possibility to achieve satisfied results using test implementations of the architecture. The implementation of the various aspects in the architecture concept will demonstrate the implementation possibility and validate the feature applicability. The objective is not to develop a complete executing architecture, but rather to gain experience and knowledge about the concept by setting the various features coming from the definite requirements in a context related to entirety.

The projects work has resulted in four main architectures: the basic architecture, the mobility architecture, dynamic configuration architecture and the adaptive service architecture. These four architectures denote together the TAPAS Platform, which requires a support system for software development, deployment, execution and management. The generic user functionality is also required in order to enable the flexibility features of the system

Ever since the project's beginning, several PhD and Master students have studied different aspects of the TAPAS concept in theses and projects. A number of publications have been produced by the project's participants, which have brought solutions and propositions to the existing architecture.

1.2 Approach

The task of this project assignment comprises a concept related to managing network shared printers called Intelligent Printing Management (IPM). This project aims to model an IPM system as an application of the developed framework in TAPAS and to show the use of Plug-and-Play.

A basic understanding of the TAPAS concept was necessary, before any work with the intelligent printing system could start. Quite an effort was spent on studying the TAPAS concepts with a particular attention paid on the TAPAS Dynamic Configuration Architecture.

Related technologies have also been studied in order to get an overall perceptive to the aspects that need to be considered before describing the system. This focus has been on complexity issues and functional requirements for printing management systems, on the network printing process and of the purpose of page-description languages in the printing process.

After having studying sufficient background information, the system description could finally commence. The focus has been on the requisite functionality for an intelligent solution, which treats the system's configuration and reconfiguration aspects. In addition, different role behaviours had to be defined in order to manage and handle the different print jobs.

1.3 Demarcations

This report has no intention of giving the reader a full introduction to all concepts of the TAPAS architecture. The focus is on dynamic configuration and reconfiguration of service components.

Implementation of the system with XML specifications and description of all the role behaviours have been taken with reservation, since the amount of work with this project is difficult to predict. The points just mentioned will be omitted if the amount of work should turn out to be more that expected.

1.4 Structure

Section 2 gives an overview of related technologies to printing in general. Different types of printer interfaces will be introduced along with the explanation of page-description languages. The two most used formats, PostScript and Portable Document Format, will be described more in detail. The network printing process will also be described, before this section ends with a study of complexity issues and general requirements for current printing management systems.

Section 3 introduces the TAPAS concept with a closer scrutiny of the TAPAS Basic Architecture.

Section 4 presents the TAPAS Dynamic Configuration Architecture with its main concepts and entities for the architectural framework. The dynamic configuration and reconfiguration aspects will also be discussed, before this section ends with a short introduction to the data model languages used.

Section 5 presents a detailed description of the Intelligent Printing Management System together with its functionality and architectural overview. Different roles are described systematically and in detail with various data models. This section ends with the achievements from this project.

Section 6 discusses the work and the results presented related to the goals of this project.

Section 7 concludes the report by summarizing the overall results and achievements. Some proposals for future work are also presented.

2 Related Technologies and Trends

This section will introduce some technologies and trends that are related to the project assignment in particular, and to TAPAS. It was important to get an overall perceptive to all the aspects that needed to be considered, before being able to describe the system. The following sections will talk about different types of printer interfaces, page-description languages, network printing, complexity issues and general requirements for printing management systems.

2.1 Printer

Printer is by definition from Columbia Encyclopedia [2] "a computer output device that reproduces data on paper or another medium". Printers fall into at least three main varieties: laser printers, ink-jet printers and dot-matrix printers. Dot-matrix printers are the least expensive, but also offer the lowest quality. Laser-printers are the most expensive printers, but also offer the highest quality. Ink-jet printers produce output almost as nice as the lower-end laser printers, and they are often less expensive.

2.1.1 Dot-matrix printer

The dot-matrix printer is a type of printer with a computer-driven, multi-pin print element (print-head), which creates characters by firing pins that strike an ink-impregnated ribbon in turn making dots on the paper. Therefore we can say that characters are made up of dots. Pins are contained in the "print-head," which moves across the paper, printing a line at a time. The printer may be a serial printer (printing one character at a time) or a line printer (assembling an entire line before it is printed). Dot-matrix printers also introduced the ability to print raster graphics. A raster graphics image, or bitmap, is a data file or structure that consists of a generally rectangular array of pixels, or points of colour, on a computer monitor, paper, or other display device. Each pixel has a corresponding red, green, and blue value that combine to determine the colour displayed by that pixel.

2.1.2 Laser printer

The laser printer is a type of printer that utilizes a laser beam to produce an image on a drum. It does this by using data sent from a computer to turn a laser beam on and off rapidly as it scans a charged drum. The drum then attracts toner powder to the areas not exposed to the light. Finally, the toner is fused to paper over a belt by heated rollers. Laser printers produce very high-quality print and are very adept at printing graphics. Because laser printers are non impact printers, they are much quieter than dot-matrix printers. They are also relatively fast, printing from about 4 to 25 pages of text per minute (ppm).

2.1.3 Ink-Jet printer

Ink-Jet printer is a type of printer that produces characters by moving the print head across the paper, projecting electrically charged droplets of ink. Because the ink is contained in replaceable cartridges, ink-jets can print in colour. Resolution approaches laser printers, and they are nearly silent.

2.2 Plotter

A plotter is a vector graphics printing device that connects to a computer. Plotters print their output by moving a pen across the surface of a piece of paper. This means that plotters are restricted to line art, rather than raster graphics as with other printers. Plotters are used primarily in drafting and computer-aided design applications, where they have the advantage of working on very large paper sizes while maintaining high resolution, but they are relatively slow because of the mechanical movement of the pens.

Another difference between plotters and printers is that a printer is aimed primarily at printing text. This makes it fairly easy to control, simply sending the text to the printer is usually enough to generate a page of output. This is not the case of the line art on a plotter, where a number of printer control languages were created to send more detailed information like "draw a line from here to here". Early plotters were created by attaching ball-point pens to drafting pantographs and driving the machines with motors controlled by the computer. This had the disadvantage of being somewhat slow to move, as well as requiring floor space equal to the size of the paper. Later versions worked by placing the paper over a roller which moved the paper back and forth for X motion, while the pen moved back and forth on a single arm for Y motion. Another change was the addition of an electrically-controlled clamp to hold the pens, which allowed them to be changed and thus create multi-coloured output.

2.3 Page Description Language

A page-description language (PDL) is a computer language that describes the text and graphics in a document. This section will give a quick introduction to PDL and its concepts. Then the two most used formats will be described more in detail (PostScript section 2.4 and PDF section 2.5). There are only a few PDLs that are in widespread use today. The most common page-description languages today are shown in Table 1 [1].

PDL name	**Abbreviation**	**Invented by**
Portable Document Format	PDF	Adobe Systems Incorporated
PostScript	PS	Adobe Systems Incorporated
Printer Control Language	PCL	Hewlett-Packard Corporation
Advanced Function Presentation	AFP	IBM

Table 1: Page Description Languages

These are the most common page-description languages used today. The table lists up the PDL names, their abbreviations and who invented them.

Today's Modern PDLs describe page elements as geometrical objects, such as arcs, lines, orb and so on. PDLs define page elements independently of printer technology, so that a page's appearance should be consistent regardless of the specific printer used. The printer itself (rather than the user's computer) processes much of the graphical information. For example, the printer carries out a command to draw a square or a character directly rather than downloading the actual bits that make up the image of the square or the character from the computer.

When printing a document on paper we usually use AFP, PS, or PCL as PDL. PDF is used to view a file on the screen whether we chose a Web browser that is capable of displaying PDF files, the Adobe Acrobat software or some other software application that allows us to view PDF files. It is however also possible to print PDF files on paper, therefore more and more people are generating PDF output (as opposed to AFP, PCL, or PS output) for printing hard-copy documents. Printers and RIPs understand page-description languages. RIP stands for Raster Image Processor, and it is a device that converts (rasteriz-ation) page-description-language code to the format required by the print engine in a printer.

There are many possible ways to use page-description languages. The following sections (2.3.1 and 2.3.2) will describe two of the most common examples.

2.3.1 Example #1:

A person named Charlize uses word-processing software to write a letter. She decides to use a laser printer to print the letter on paper, and mails it via U.S. Mail, as represented in Figure 1.

Let us assume that Charlize is using Microsoft Word to write the letter. After she has finished writing it, she wishes to print it out. Microsoft Word then generates page-description-language code that describes the contents of the letter; and her computer sends the page-description-language code to the RIP inside her laser printer. (Depending on her printer, the page-description-language code will probably be either PS code or PCL code.) The RIP rasterizes the page-description-language code and sends the result of the rasterization process to the print engine in her printer. The print engine prints the letter by using toner or ink to draw text and graphics (if any) on the paper.

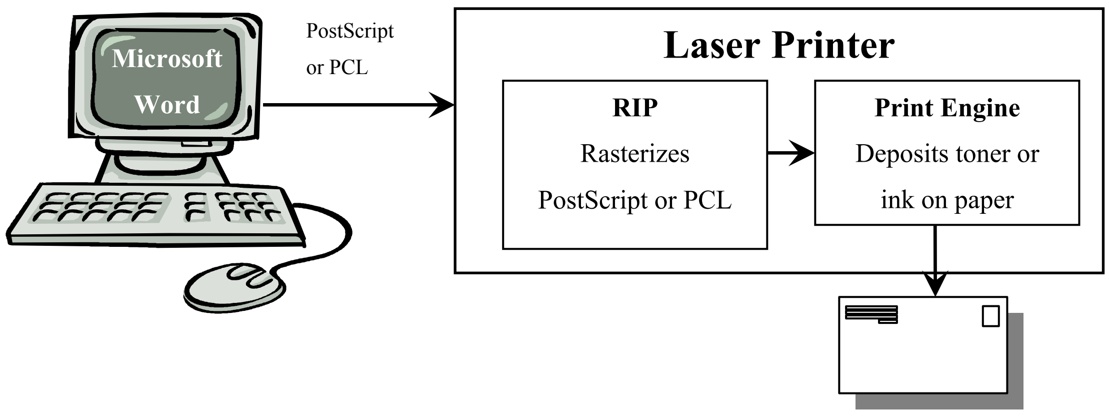


Figure 1: Page-description language example #1.

The RIP rasterizes the page-description-language code and sends the result of the rasterization process to the printer's print engine.

2.3.2 Example #2:

A person named Bill uses page-layout software to create a newsletter. Then he converts it to PDF format and distributes it via e-mail as depicted in Figure 2.

Let us assume he is using QuarkXPress to create the newsletter. After he has created it, he instructs QuarkXPress to make a PS file containing a description of the text and graphics in the newsletter. Then he uses Adobe Acrobat Distiller to convert the PS file to PDF format, and he e-mails the PDF file to each person on the distribution list for the newsletter. Some recipients of the newsletter simply use to open the PDF file and to read it on the computer screen. Others prefer to read the newsletter on paper and they print it out. The Adobe Acrobat/Adobe Acrobat Reader software generates page-description-language code (probably either PS code or PCL code, depending on the printer). The computer sends the page-description-language code to the RIP inside the printer. The RIP rasterizes the code and sends the result of the rasterization to the printer's print engine, which prints the text and graphics by depositing toner on the paper. The software applications that he uses to create and print his documents will automatically generate page-description language code that is appropriate for the printer he is using.

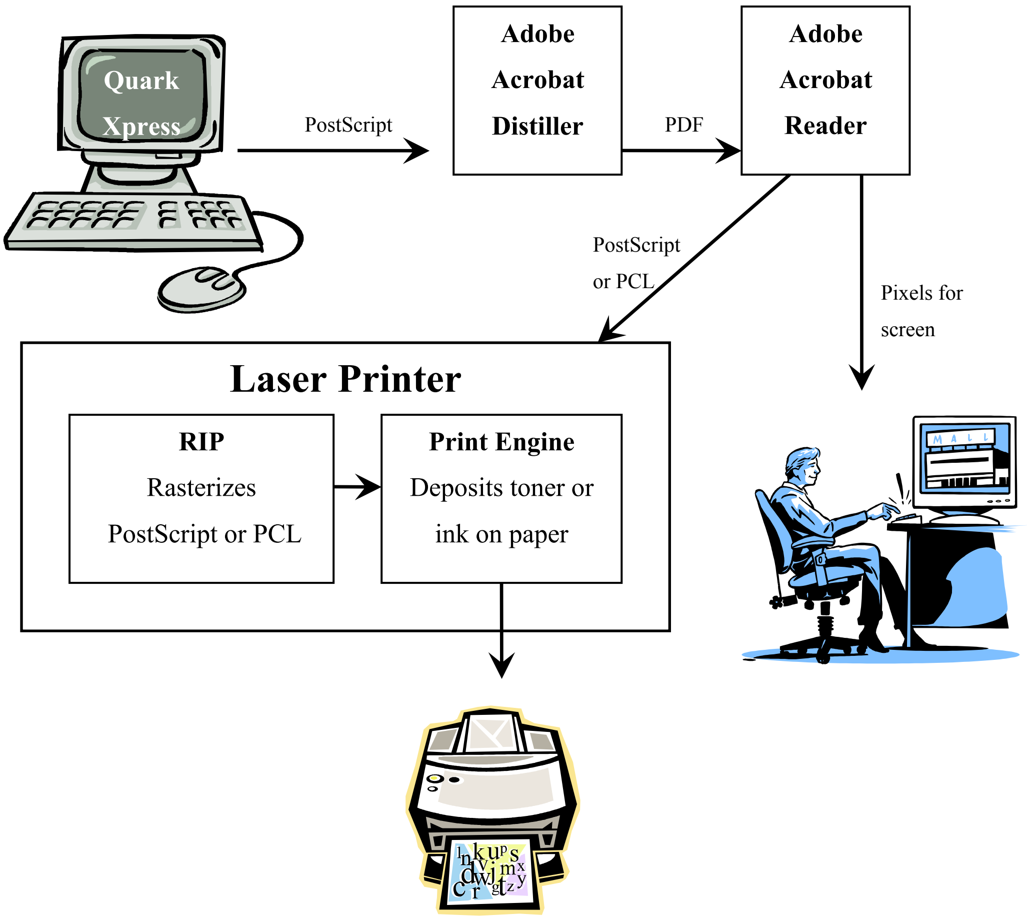


Figure 2: Page-description language example #2.

2.4 PostScript

PostScript (PS) was invented by Adobe Systems Incorporated and was once the de-facto standard for electronic distribution of final documents, though it has effectively been succeeded by PDF in this area. This section will quickly describe PS as a page-description language and as a programming language.

The PostScript specification [3] states:

"*THE POSTSCRIPT® LANGUAGE is a simple interpretive programming language with powerful graphics capabilities. Its primary application is to describe the appearance of text, graphical shapes, and sampled images on printed or displayed pages, according to the Adobe imaging model. A program in this language can communicate a description of a document from a composition system to a printing system or control the appearance of text and graphics on a display. The description is high-level and device-independent.*"

PostScript (file extension .ps) is a page-description and a programming language designed to do one thing: describe extremely accurate what a page looks like and how to display text or graphics on a printed page. PS broke tradition by combining the best features of both printers and plotters. Like plotters, PS offered a high quality line art and a single control language that could be used across any brand of printer. Like dot-matrix printers, PS offered simple ways to generate pages of text and raster graphics. Unlike either, PS could place all of these types of media on a single page, which offered far more flexibility than any printer or plotter previously had. Every programming language needs a processor to run or execute the code. In the case of PS, this processor as mentioned earlier is a combination of software and hardware which typically lives in a printer, and we call it a Raster Image Processor (RIP). RIP takes in PS code and renders it into dots on a page. So a PS printer is a device that reads and interprets PS programs, producing graphical information that gets imaged to paper, film, or plate.

2.4.1 PostScript as a page-description language

PS has a large selection of graphics operators that allow it to precisely describe a desired page. These operators control the placement of three types of graphics:

- **Text** in a wide variety of typefaces can be placed on a page in any position, orientation, and scale.

- **Geometric figures** can be constructed using PS graphics operators. These describe the locations of straight lines and curves of any size, orientation, and width, as well as filled spaces of any size, shape and colour.

- **Sampled Images** of digitized photographs, free-hand sketches, or any other image may be placed on a page in any scale or orientation.

All graphic objects may be easily rotated, scaled, and clipped to a specified portion of the output page.

2.4.1.1 PostScript Imaging Model

A high-level *imaging model* enables an application to describe the appearance of pages containing text, graphical shapes, and sampled images in terms of abstract graphical elements rather than in terms of device pixels. The PS model considers an image to be built up by placing ink on a page in selected areas. The ink may form letters, lines, filled shapes, or halftone representations of photographs. The ink itself may be black, white, coloured, or any shade of gray. These elements may be cropped to a boundary of any shape as they are placed on the page. Once the page has been built up to the desired form, it may be printed on an output device. Such a description is economical and *device-independent*.

A page-description language is a language for expressing an imaging model. An application program produces printed output through a two-stage process:

1.) The application generates a device-independent description of the desired output in the page-description language.
2.) A program controlling a specific raster output device interprets the description and renders it on that device.

The two stages may be executed in different places and at different times; the page-description language serves as an interchange standard for transmission and storage of printable or displayable documents.

2.4.2 PostScript as a programming language

About one-third of the PS language is devoted to graphics. The remainder makes up an entirely general computer programming language. The PS language is an interpreted, stack-based language similar to Forth. Reading a program requires some practice since one has to keep the layout of the stack in mind.

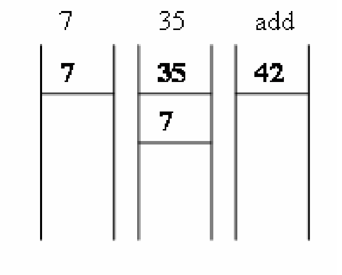
2.4.2.1 The PostScript Stack

PS reserves a piece of memory called a stack for the data with which it is working. The stack behaves like a stack of books. This means that the last book put on the stack is the first book that can be taken off. Similarly, numbers, strings, and other pieces of data placed on the stack will be removed in reverse order, the last item added to the stack being the first retrieved. This type of data structure is referred to as a last in, first out or LIFO stack.

2.4.2.2 Postfix Notation

PS operators that require numbers or other data, such as add and sub, retrieve that data from the stack. To use an operator, one must first place the data it requires, its operands, on the stack, and then call the operator. The operator will place its own results on the stack. In this notation, the operands precede the operator, thus dispensing with the need for parentheses. This style of programming is referred to as postfix notation.

The add operator causes PS to remove the top two numbers from the stack, add them, and leave the sum on the stack. Thus, the program line below would affect the stack as illustrated in Figure 3. The 7 and the 35 are pushed onto the stack and the add operator then replaces them with their sum.



7 35 add

Figure 3: Stack operations

The figure shows how operands are being pushed on the stack and how the operator then replaces the result on the stack.

2.4.2.3 PostScript Data Types

PS supports many data types common to other languages, including reals, booleans, arrays, and strings. The PS language also defines object types such as dictionary and mark.

2.4.2.4 PostScript Flexibility

PS is an extremely flexible language. Functions that do not exist, but which would be useful for an application, can be defined and then used like other PS operators. Thus, PS is not a fixed tool within whose limits an application must be written, but is an environment that can be changed to match the task at hand.

2.4.2.5 Printable Programs

PS programs are written entirely in printable ASCII characters. This allows them to be handled as ordinary text files by the vast majority of communication and computer file systems. In addition, it ensures that a PS program will be as easy for a person to read as the structure of the program allows. The PS programming language, like all programming languages, works with various types of data, such as numbers, arrays, strings, and characters. The pieces of data manipulated by PS are referred to as PS objects.

2.5 Portable Document Format

References [4], [5]

The PDF has quickly grown to be a widely-used format for capturing and exchanging formatted documents electronically, across the Web, via e-mail and for virtually every other document exchange mechanism. This chapter will give a quick introduction to PDF and explain more in detail what the main differences between PS and PDF are.

2.5.1 Introduction to PDF

PDF (file extension .pdf) is a file format developed by Adobe Systems for representing documents in a manner that is independent of the original application software, hardware and operating system used to create those documents. A PDF file can describe documents containing any combination of text, graphics, and images in a device independent and resolution independent format. These documents can be one page or thousands of pages, very simple or extremely complex with a rich use of fonts, graphics, colour and images.

2.5.2 Short History

When PDF first came out, in 1993, it was slow to catch on. At the time, not only did the only PDF creation tools of the time (Acrobat) cost money, but so did the software to view and print PDF files. Additionally, there were competing formats. Adobe started distributing the Acrobat Reader program at no cost, while competing formats eventually died out and PDF became a well-accepted standard.

2.5.3 Technology

PDF relies on the same imaging model as the PS page description language to render complex text, images, and graphics in a device and resolution-independent manner, bringing this feature to the screen as well as the printer. To improve performance for interactive viewing, PDF defines a more structured format than that used by most PS language programs. PDF is primarily the combination of three technologies:

1. a cut-down form of PS for generating the layout and graphics
2. a font-embedding/replacement system to allow fonts to travel with the documents
3. a structured storage system to bundle these elements into a single file, with data compression where appropriate

PDF also includes objects, such as hypertext links and annotations that are not part of the page itself, but are useful for building collections of related documents, for reviewing and commenting on documents. The special feature about PDF is that it contains only those PS language elements that define the graphics and that require a very simple interpreter. This means that the process of turning PDF back into a graphic is a matter of simply reading the description, rather than running a program in the PS interpreter. However the entire PS world in terms of fonts, layout and measurement remains intact.

The PS-like PDF code is often generated from a source PS file. The graphics commands that the PS code outputs are collected and tokenized, any files, graphics or fonts the document references are also collected, and then everything is compressed into a single file.

2.5.4 Differences between PostScript and PDF

I want to end this chapter by describing some important differences between PS and PDF, giving you some reasons to understand why PDF is becoming a replacement for PS in many situations. PS and PDF are quite similar, where each is a file format that describes text and graphics, but there is more to it than that. PS was designed to describe a page. PDF does this as well, but beyond this, PDF can also contain information not only related to how a page looks, but also can describe how it behaves and what kind of information is contained in the file.

The most important difference between PS and PDF is that it is easy for most computer users to view PDF files on the screen, and rather difficult or impossible to view PS files on the screen. This is because it is easier to obtain good-quality PDF-viewing software (like Adobe Acrobat Reader) free of charge, than to obtain PS-viewing software.

Another difference as mentioned earlier between PS and PDF is that PS is a programming language, but PDF is not. PS is a programming language because it works with various types of data, such as numbers, arrays, strings, characters. The programming language also has constructs (e.g. conditional constructs and looping constructs) associated with it. PDF does not have these things.

In general, a PDF file that describes a particular document is smaller than a PS file describing the same document. This is an important property, because smaller files can be handled more efficiently than larger ones (it takes less time to send smaller files across a network and smaller files take up less space on your computer's disk).

At last, PDF can do lots of things that PS cannot do. PDF files can be viewed on the Web (with the proper software), whereas PS files normally cannot. A PDF file can contain links to locations within the same PDF file, within other PDF files, or on the Web. A PS file can normally not contain links. A PDF file can function as a data-entry form, but a PS file cannon.

2.6 *Network Printing*

Printing is generally divided into two cases, direct printing (non-network printing) and network printing. This section will introduce both of them, but with a closer view on network printing.

2.6.1 Direct printing

For most home-users, printing may seem fairly simple since the printer is cabled directly to the computer. A document is written and the data is sent directly to the printer as shown in Figure 4.

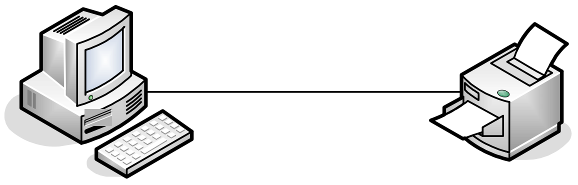


Figure 4: Direct printing

Direct printing is the instance when a printer is cabled directly to the computer.

The steps in a non-network printing procedure are storage, intermediate processing and transmission between various processing locations [6].

2.6.2 Steps in the network printing process

Printing is one of the major reasons companies turn to networking, because they need to operate more efficiently, e.g. the need to share printers among several users. Network printing is most common in organizations and companies that install large multifunctional printers and in small home offices that wish to share a printer among family members. Several intervening steps must take place when printers are shared and each step in the printing process affects the time necessary for a print job to be completed at the printer. The typical path print data might follow during the printing process is illustrated with the following scenario, as showed in Figure 5.

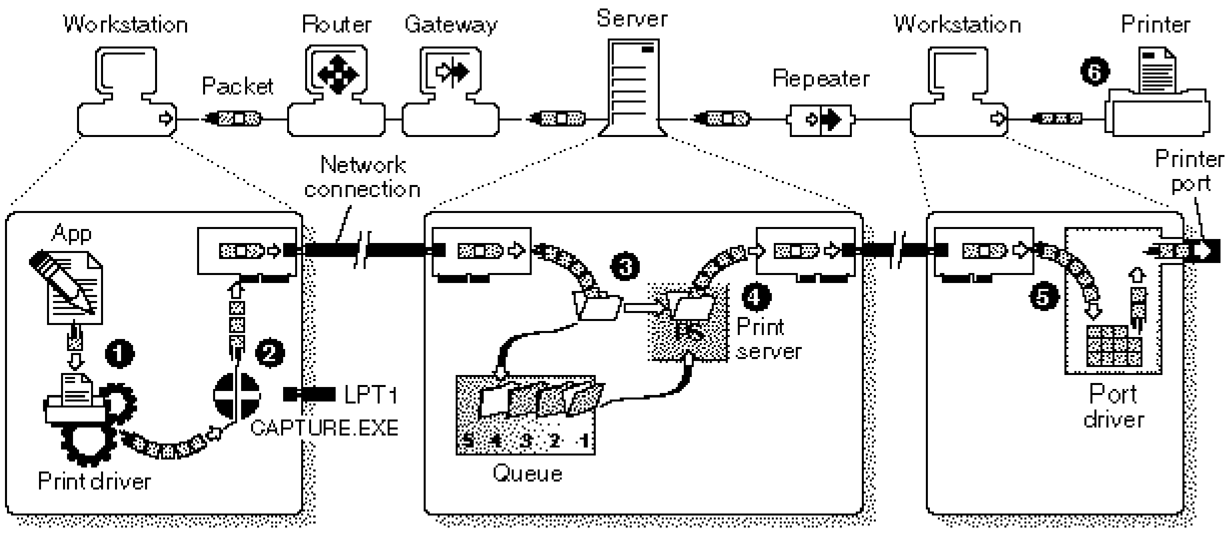


Figure 5: A typical network printing setup [6]

2.6.2.1 Step 1: Print Data Is Generated and Transmitted

The application compiles the data entered by the user and passes it on to a print driver. A print driver generates the actual printer data in a page-description language code (either PS or PCL depending on the printer) and passes it toward LPT1. The driver may be part of the application software, or may have been supplied with the operating system, or may have come from a third party. The speed at which print data can be generated decreases with the amount of additional formatting and graphical requirements. Streams of simple text characters are created almost instantaneously on modern systems. Formatted text, which requires precise alignment of varying fonts and sizes, produces higher quality output but is more complex and takes longer to generate.

2.6.2.2 Step 2: Data Is Redirected to a Network Queue

In non-network printing, data generation and transmission can be nearly simultaneous on a single computer. However, during network printing, data is not sent directly to a printer as it is in non-network printing. Instead, the data is redirected to a file in a print queue where it is stored while waiting to be sent to a printer. On its way to the print queue, the data is assembled into small packets, which are labelled and passed to a network interface board and the associated driver software. Along the way over the network medium towards the network interface, the individual packets have to traverse repeaters, routers, gateways, and bridges.

2.6.2.3 Step 3: Data Is Stored in a Print Server Queue

Once arrived at the network interface, each packet is transmitted over to the destination file server that will store the data, check the packets for transmission errors and send acknowledgements back to the sending interface. The data packets are stripped of their labelling header and stored as a file on the server's hard drive. When the data for a complete job has been received and stored, the file is closed and the filename is added to the queue associated with the printer. The stored print job waits in the queue until the print server is able to transmit it to a printer. If a print job is behind another job in the queue, the associated print server will request and receive the data file information immediately upon completion of the preceding job. The delay can be long if there are many other print jobs waiting in the print queue, or if the user stipulated that this job should print at some particular time in the future.

2.6.2.4 Step 4: Print Data Is Transmitted to a Printer Station

After the print job information is stored in the print server, it's now ready to be executed. The print server starts reading the print data from the queue by assembling the data into small packets, labelling each with a header and then passing them to a network interface board and associated software. As the packets are sent, the print server gets packets back from the printer station containing information about the status of the printer, the amount of data that has actually been sent to the printer, and how many packets the print station currently has room to receive.

2.6.2.5 Step 5: Print Data Is Transmitted to the Printer, which Formats the Data and Completes the Print Job.

Once arrived at the workstation where the printer is attached to (printers are not always attached to a workstation), the packets labels are removed and the print data is passed to the port driver. The printer port is initialized once the port driver is loaded. If the printer is not busy, one character is sent. Data travels by cable to the printer where it is stored until enough data has been accumulated and converted to complete a single physical cycle. The cycle for a laser printer is a full page, while for most other printers the physical cycle is on pass of a print head. The formatting process can be brief or lengthy depending on the type of data being processed. During this process, the computer communicates with the printer, waiting for a ready signal between each byte that is sent. If the printer has insufficient memory to store new data until the physical cycle has finished with the old data, the printer may signal the port driver to suspend transmission during each cycle. When the last physical cycle of the printer is complete, the print job is done.

2.7 Complexity Issues

Before describing the functionality of the IPM system, it is important to look at the complexity issues for today's printing management system (PMS). Studying when and where current PMS fail or reach their limits, what the problems are due to and what deficiencies can be improved will be of much help when specifying the requirements, describing the idea, the solution and the architecture for the IPM system.

2.7.1 Multi-protocol servers

One major problem to start with, would be to look at the fact that there exist many different network operating systems and that companies and organizations use different platforms. Printing problems at large organizations frequently occur because both the direct printing and the server-based methods are used. Usually, PC users print to Windows print queues, Unix users to Solaris or Unix Print Services, and Macintosh users through AppleTalk. Each client operating system is given a "central" server, and multiple print servers contend with each other for access to the printer, which most often destroys orderly queuing and leads to unpredictable delays. Some printers have implemented queuing services in different printer hardware in order to solve this problem, but printers tend to have coping problems when multiple simultaneous jobs arrive on different protocols. The printer can find it difficult to switch between different protocol suites and could therefore lock up when a Unix job is sent following an AppleTalk job.

IT department use a lot resources to deploy multiple printer servers, this because they often involve multiple teams and solving printing problems on different server platforms is quite complex. Problems with print service tend to be duplicated on each server platform and must be solved on several different platforms instead of just one. Problems lie habitually in the network and fixes to one protocol suite tend to cause problems for other protocols, and it is not common to maintain staff that has deep expertise in all the protocol suites used.

The solution would be to replace these multiple print servers with a multi-protocol server conversant with all the network client protocol suites as well as any protocol used to relay jobs from the server to the printers. Operating with only one server platform results in queues that are orderly maintained and staff time is saved because only one server platform calls for troubleshooting and maintenance. Printers are suddenly made more accessible, since multi-protocol servers translate between different protocols, and jobs submitted by Macintosh clients may now be sent to a Windows printer.

2.7.2 Bi-directional feedback between printers and the PMS

The bi-directional feedback relationship between PMS and printers permits saving time and resources when managing printers. It allows clients and printers to exchange information about printers and print jobs, and for clients to get real-time information about a printer, such as its availability, its status, and what its configuration properties and features are. Clients get information such as “printer offline”, “printer out of paper”, “service requested”, “no toner”, “door open” or “paper jam”, and this information may be sent to users via email, a network message, pop-up screen, log file or paper. These messages show some of the most common problems we find today with any kind of printer, and by being notified when a printer is in trouble or is about to be will ease the printing management and reduce the printer’s downtime. Remote-control makes it easy to manipulate the printer control panel from the administrators own desktop as if he was physically standing in front of it, but this feature depends on the type of printer and vendor.

2.7.3 Printing in Letter format

Some printers may have the feature of printing Letter format, which is not equivalent with the standard A4 format that most printers use. When a Letter print job is sent on a printer that supports A4 formats, the printer stops since it received a print format that it does not recognize. To solve this problem, one has to manually tell the printer to print out the file in A4 format by holding down 2 buttons on the printer hardware. This entails large print job queues if the downtime of a printer is long. This type of problem can easily be avoided if the PMS could notify the client, before the print job is sent to the printer, that its print job is not compatible with the selected printer.

2.7.4 Build in fault tolerance with PMS

Just as a person is about to send an important report to the printer, half the network goes down and that specialized, high-density colour printer the user needs is no longer available. The print job might be lost if it was already sent. Printer objects could be restored if the PMS had a distributed replicated directory that does not depend on any server or group of servers to function. An object from a crashed server can be "hot loaded" onto a functioning server, and printing proceeds after a brief interruption.

2.8 General Requirements for PMS

Studies produced by IDC (2000) and Cap Ventures, Inc. (2001) show that IT staff spends 15 percent of its time on printing-related issues and that most IT managers are unaware of how much they are spending on their printing and imaging environment. Printing Management Systems make print network more efficient, easier to manage, more scalable, and more available to users. It also let enterprises save a lot of money on printing costs by combining powerful management tools, reliable infrastructure, unparalleled connectivity, intelligent printers and multifunction products. Printing Management Systems is a smart solution for small workgroups to enterprise-wide systems who wish to eliminate costs associated with printing and who wish to have totally control when managing network printing. This section examines some requirements for Printing Management Systems that are important to highlight, and that will be of great help when describing and elucidating the solution and the functional requirements for the IPM System.

2.8.1 IPP and Internet Printing

Source: [9]

Internet Printing is the process of sending a print job request directly to a printer that has an IP address via the Internet. Internet Printing is possible due to the Internet Printing Protocol (IPP), an application level protocol that can be used for distributed printing using Internet tools and technologies and that covers most common situations for printing on the Internet. To ensure Internet Printing, printers need to support the Internet Printing Protocol (IPP). The printer can be connected to the Internet either outside or inside a firewall, but with different approaches. If the printer is connected to the Internet outside the firewall, some settings on the network router may need to be changed to allow communication with the printer from the

Internet. Normally, external sources need to communicate with the printer using the HTTP protocol on ports 631 and/or 80. When the printer is connected to the Internet inside the firewall, it is important so ensure that the necessary communication ports and protocols are enabled for the printer. For an external application to use IPP with an internal printer, it must be possible for the inbound HTTP protocol to access the printer. The IPP requests ports 631, 80 and 8080.

2.8.2 Remote-control installation and configuration

Remote-control installation, allows network administrators to install printers on selected desktops from their own console. The next time the user logs on, the printer is automatically installed, printer driver and all. Remote-control makes it easy to manipulate the printer control panel from the administrators own desktop. It allows them to centrally manage their networked printers as if they were physically standing in front of it, which decreases the time they spend travelling to and from the printers or to end-user workstations.

2.8.3 Event notification

Event notification allows administrators to specify interested parties who should be notified of an event or problem and how they should be notified. For example, notification can be configured so that the owner of a print job will receive a pop-up screen message when the job has actually been printed. An administrator can also ensure that a printer operator is notified when a printer problem occurs, such as paper jam, no toner, door open or printer out of paper. Other notification methods should also be included, such as email, network message and log file records. There are some problems that can be avoided with the use of pre-emptive failure notification with alert messaging, even when the printer is printing, e.g. the printer senses when toner is low or trays are about to run out of paper, and alerts administrators with one of the notification methods just mentioned. Most failures occur during printing, and this feature prevents unnecessary delays and ensures maximum production efficiency.

2.8.4 Plug-and-Print

Installing new printer hardware should require as little configuration as possible. Some vendors have created gateways that easily install public-access printers, and which requires no further intervention from the user after hooking the printer up to the network cabling. Network-able printers are configured to communicate location and status to the print server through these gateways. Just Plug and Print.

2.8.5 Move jobs on the fly

Be able to move a print job from a busy printer to a less-busy printer without having to delete and resend it. If a user has the misfortune of sending a print job just right after a 1023 MB, he could spend half the day waiting for his document to emerge. Users can access the queue of the printer to check how many jobs are ahead of them before they send their document. The users can also get notified with the event notification when their print job has finished, so they do not have to hang around the printer and wait for their document to emerge.

2.8.6 Multi-protocol servers

Networks run on different platforms, each requiring their own separate management infrastructure. Current Printing Management Systems will have to support the different protocols that exist, and one solution would be a multi-protocol server conversant with all the network client protocol suites as well as any protocol used to relay jobs from the server to the printers. Operating with only one server platform results in queues that are orderly maintained and staff time is saved because only one server platform calls for troubleshooting and maintenance. Printers are suddenly made more accessible, since multi-protocol servers translate between different protocols.

3 TAPAS Basic Architecture

The TAPAS project has developed four different architectures [16], which are denoted as the TAPAS Platform. The different architectures are shown in Table 2. This section will take a closer look on the Basic Architecture of TAPAS, which defines a service system as a decomposition of service components. In addition will concepts like dynamic plug-and-play (3.1), the theatre metaphor model (3.2), and the basic architecture object model (3.3) be presented in the following subsections. The Basic Architecture has already been implemented using Java RMI technology [7].

Architecture	Description
TAPAS Basic Architecture	The basis for all dynamic behaviour functionality. Defines a service system as a decomposition of service components.
TAPAS Dynamic Configuration Architecture	Based on TAPAS basic architecture. Provides support for dynamic configuration and reconfiguration of systems.
TAPAS Mobility Architecture	Based on TAPAS basic architecture. Provides different types of mobility.
TAPAS Adaptive Service Architecture	Handles complexity and diversity issues in a global-scale. Demands an infrastructure which provides interfaces and enables dynamic interoperability of services regardless of their programming languages and operating environments.

Table 2: TAPAS Platform

The table shows the four different platforms of the TAPAS architecture [16].

3.1 Definition of Plug-And-Play

References: [17]

Plug-and-play (PaP) is a standard that gives computer users the ability to plug a device into a computer and have the device recognized automatically. This function is performed by the operating system which makes the installation of the product quick and easy. Network elements have the ability to configure themselves when installed into a network (to plug) and then to provide services (to play) according to their own capabilities. There exist two kinds of PaP. *Static PaP* is when you plug-in a network component and the system simply work since the component ant the framework has predefined functionality.

The TAPAS project uses a more general kind of PaP called *dynamic PaP.* In this kind of PaP, the plugged-in unit has a set of basic capabilities, but its functionality is defined as a part of the plug-in procedure and it can be changed dynamically. In other words, the definition of individual components as well as the structure of components can be changed on-line. Another aspect of dynamic PaP is to change the services that a component provides and to make sure that all the users have the ability to employ the services.

From now on the concept PaP means dynamic PaP.

3.2 *The Theatre Metaphor Model*

References: [16]

As depicted in Figure 6 the theatre metaphor is used to describe the concept of TAPAS. Actors perform roles according to predefined manuscripts. They have different capabilities telling which roles they are able to play. TAPAS theatres consist of plays that are managed by directors, which is a special type of actor responsible for supervising other actors.

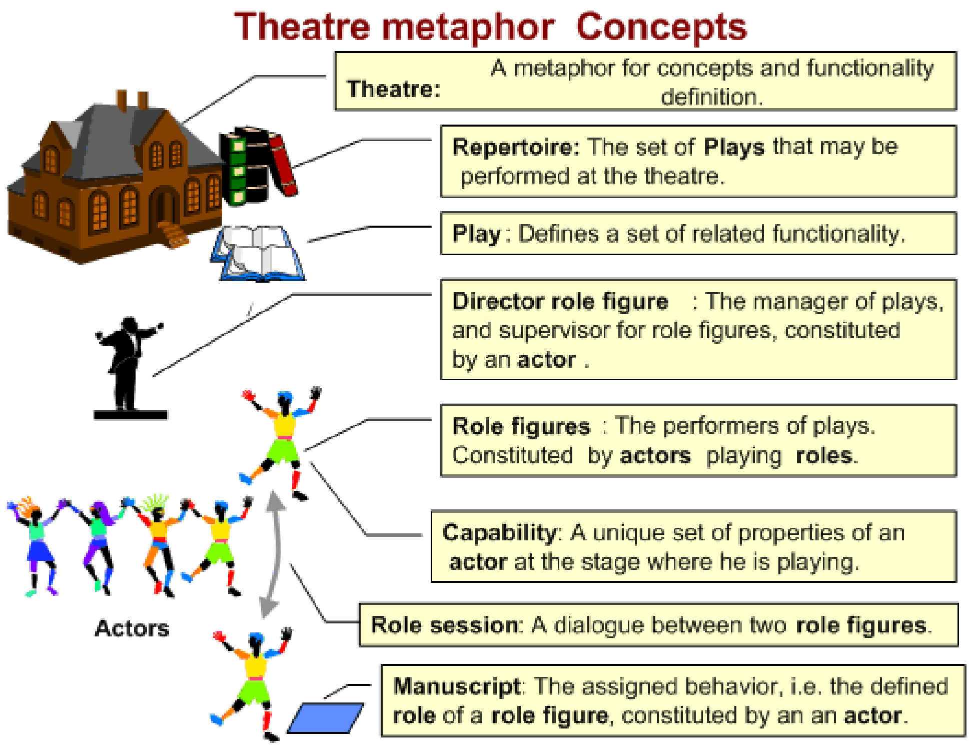

Theatre metaphor Concepts
Theatre: A metaphor for concepts and functionality definition.
Repertoire: The set of Plays that may be performed at the theatre.
Play: Defines a set of related functionality.
Director role figure : The manager of plays, and supervisor for role figures, constituted by an actor .
Role figures : The performers of plays. Constituted by actors playing roles.
Capability: A unique set of properties of an actor at the stage where he is playing.
Role session: A dialogue between two role figures.
Manuscript: The assigned behavior, i.e. the defined role of a role figure, constituted by an an actor.
Actors

Figure 6: Concepts used in the theatre analogy [16]

The figure shows the concepts used in TAPAS in order to realise PaP.

An actor may interact with another role when it performs according to a manuscript. All actions related to another role are performed as a part of a role session. The interaction with the cooperating actors is specified in the manuscripts, as how to reach the actor and what to do with an incoming interaction.

3.3 TAPAS Basic Architecture Object Model

References: [17]

The TAPAS basic architecture is illustrated by its object model in Figure 7. The gray box shows the play view of the architecture. It is based on actors in the nodes of the network that can download manuscripts defining roles to be played. An actor is the software component that executes role-figures. The model is founded on the theatre model described in section 3.2.

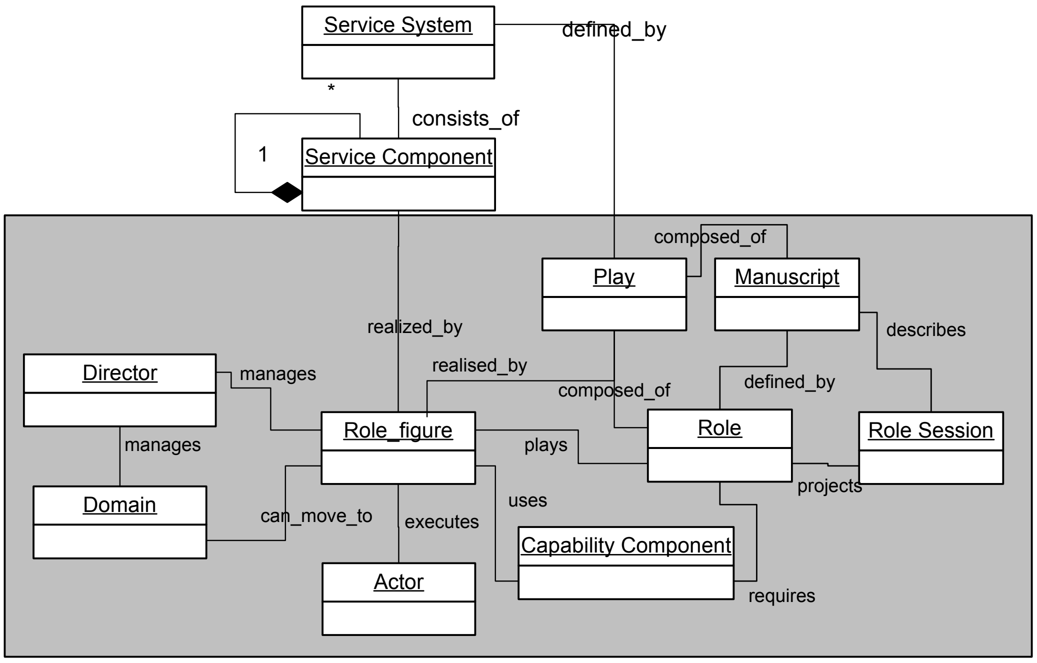


Figure 7: TAPAS basic architecture (object model) [16]

The architecture shows that a service system consists of service components with their functionality defined by plays. A play consists of a number of actors playing different roles. Each role has different requirements on capabilities and status for a given system. All actions related to another role are performed as a part of a role session. An actor comprises a role-figure by behaving in accordance with a manuscript defining the role's functional behaviour in a play. A service component is realised by a role figure based on a role defined by a manuscript. A role figure is realised in a node and employs the node's capabilities. For an actor to play a role in a node, it needs to ensure that the node's capabilities match the required capabilities.

4 TAPAS Dynamic Configuration Architecture

The TAPAS project aims to create a PaP system where software and hardware components are able to configure themselves once installed into the network and able to adapt to changes in the system environment. The dynamic configuration and reconfiguration aspects in PaP systems are the main focus for the dynamic configuration architecture. TAPAS dynamic configuration architecture is based on TAPAS basic architecture, which defines a service system as a decomposition of service components.

The following sections will describe the main concepts and entities for the architectural framework (4.1 and 4.2). The languages used in the data models will also be introduced briefly (4.3).

4.1 Capabilities and Status

Resources: [7]

The concept of capabilities and status is very important in the dynamic configuration architecture, and since it is a main part used for decision making it will be explained further in this section.

The concept of capabilities was introduced in section 3.2. Actors execute roles defined by manuscripts and where roles are realised in nodes. Actors were said to have a set of capabilities which tells what the actors are able to do. The capabilities in the dynamic configuration architecture on the other hand are said to be inherent properties of nodes, not actors.

Capabilities can be classified into different categories, as shown in Figure 8. The first category is called primitives where capabilities can be classified into resources, functions and data. Resources refer to “physical hardware components with finite capacity, such as processing, storage and communication units”. Functions refer to “pure software or combined software/hardware components which perform particular tasks”. Data refers to just data. Even though functions and resources are divided into two separate types, they are commonly related and dependable on each other. A resource may realise many functions, and a functions can utilise a variety of resources.

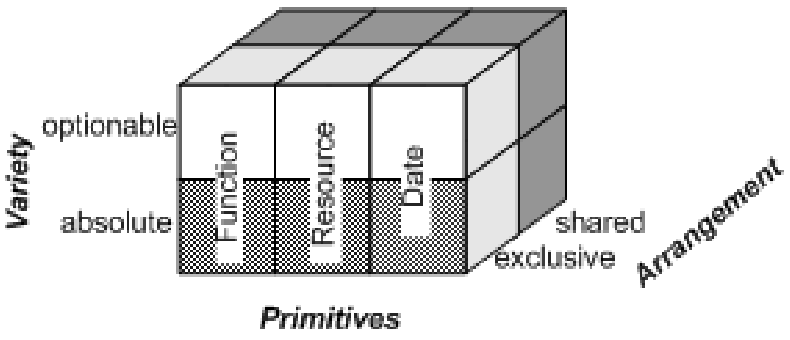


Figure 8: Classification of capabilities with their attributes in a three-dimensional view [7]

Capabilities can be classified into different types of primitives: Resources, functions and data. Capabilities can also be characterized by their variety and arrangement, i.e. whether they are optionable or absolute and whether they are shared or exclusive.

Capabilities can also be classified in other categories, such as by their varieties and by their arrangement. Capabilities can be either optionable or absolute when classified into varieties. Take for example the capability classified as a transmission function. This type of capability is optional, i.e. it has a variety of bandwidth options. Contrary to access right data, which is classified into absolute, has no option for negotiation. Furthermore, when capabilities are classified into their arrangement, they can be either exclusive or shared. A pin code is exclusive information only known by one user, while users can share their files and resources when connected onto a network.

The status in a PaP system is the situation at a certain time instant. A depiction of the situation in a system comprises the actual number of nodes, the state of each node, how many plays are actually playing, and information on the traffic situation.

4.2 Framework

Resources: [7]

The architecture framework for dynamic configuration in PaP systems is depicted in Figure 9. It consists of different entities, each having their special responsibilities and functions that are necessary to provide dynamic configuration functionality. Each entity will be described in the subsequent subsections.

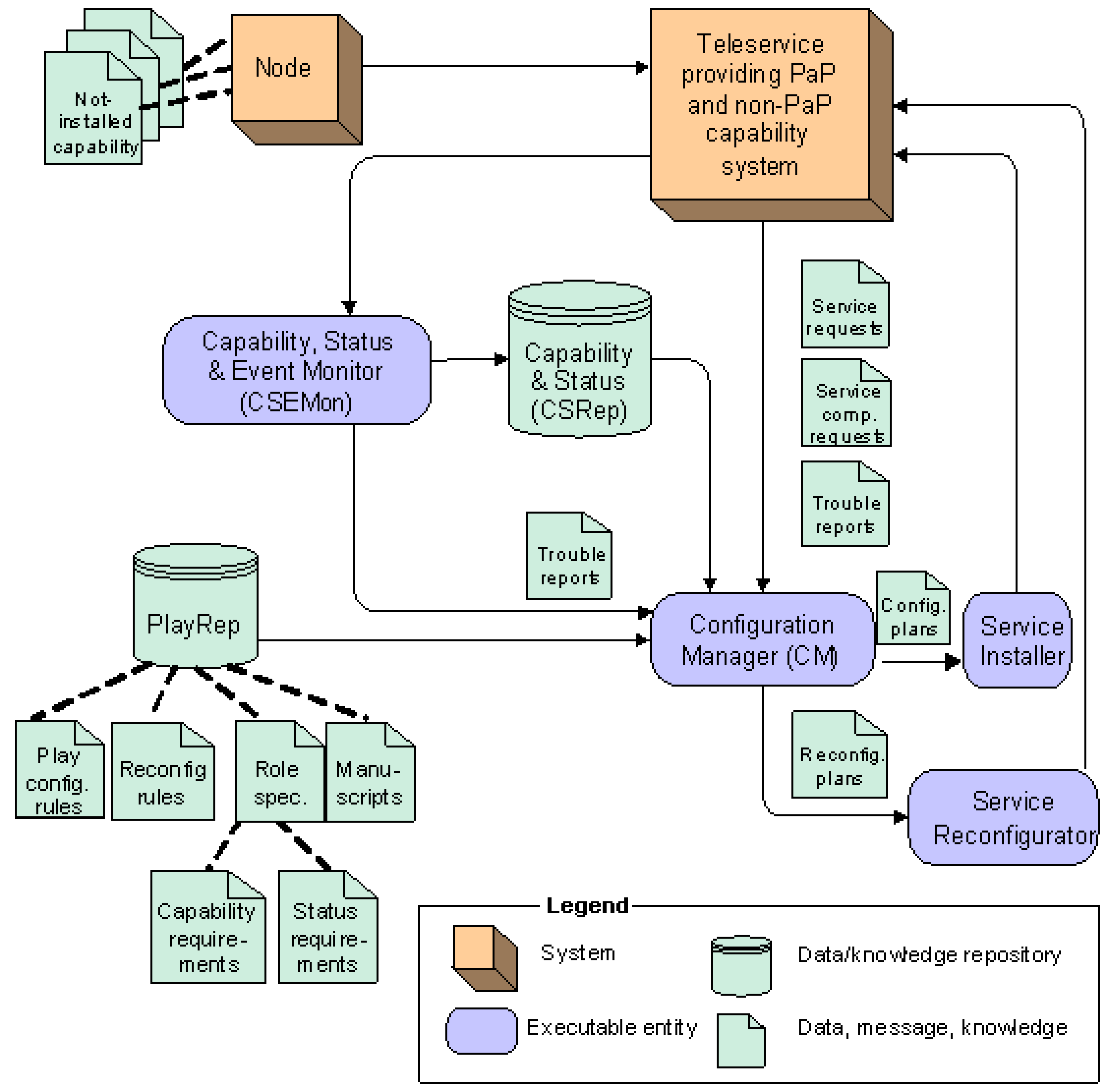


Figure 9: Architectural framework for dynamic configuration [7]

4.2.1 Capability and Status Repository

The Capability and Status Repository is the first out of two repositories that exist in the framework. This repository contains specifications of capabilities offered by components and nodes in the system, and it preserves information regarding the system's situation and status at a particular time.

4.2.2 Play Repository

The second repository included in the framework is the Play Repository (PlayRep). This repository contains a set of play definitions defining functional behaviours of particular service systems. Each system consists of actors playing different roles with certain constraints on capabilities and the status situation. A play is defined by manuscripts, where each manuscript defines behaviour of one of the roles participating in the play. Each role has role specifications that identify the role's requirements on capabilities and status. Also related with a play are play configuration rules that describe the constraints on the service system configuration. These constraints are set in order to evade an overload situation. The total number of roles allowed to install at a specific node could represent such a constraint, and e.g. the total number of actors allowed to run on the same node. The last specification associated with a play, reconfiguration rules, are policies for handling events related to reconfiguration. Events of this sort may be resource unavailability and service component failure. The system uses the reconfiguration rules to repair and reconfigure itself if it should detect an abnormal problem or a unusual system behaviour.

4.2.3 Capability, Status and Event Monitor

The Capability, Status and Event Monitor (CSEMon) supervises the capabilities and the status of the PaP system, in addition to keep the CSRep updated with new changes in the system. Furthermore, it listens to events indicating changes to the system and its environment that would prevent the system from getting the desired level of services. If it should detect such a system change, it notifies the Configuration Manager with trouble reports as depicted in Figure 9. The Configuration Manager responds to trouble reports indicating problems by producing a reconfiguring plan in order to keep the system functioning with an acceptable QoS level.

4.2.4 Configuration Manager

The Configuration Manager (CM) is responsible for handling requests for installing new devices and requests for instantiation of service components, along with taking action when error occurs. CM decides its actions based on the information stored in PlayRep and CSRep.

When a service request is sent the CM carries out the play definition from PlayRep along with system capabilities and status from CSRep. An appropriate configuration plan is then selected based on the information retrieved, which will be forwarded to and executed by the Service Installer (SI). Handling instantiation of service components, or service component request as they also are called, is quite similar. CM dynamically determines the best node to install based on, like previous, available capabilities and status from the PlayRep, together with the component's requirements from the CSRep. The SI is then notified to load a corresponding manuscript from the PlayRep and to instantiate it on the suggested node.

Upon the receipt of a trouble report the CM will analyse the indicating problem, retrieve related information from the CSRep and the PlayRep that will be used to produce a service reconfiguration plan. The content of this plan will be executed by the Service Reconfigurator (SR). The CM selects the most appropriate plan based upon the nature of the problem and on the defined reconfiguration rules.

4.2.5 Service Installer and Service Reconfigurator

The SI is responsible for installing the configuration plans created by the CM as a result of a service request. SI will create the actors specified for certain roles, in addition to certify that allocation of capabilities as well as instantiation of a manuscript for each role is achieved. Reconfiguration plans will in the same way be handed over to the SR, which initiates and performs reconfiguration based on the plan.

4.3 Introduction to Data Model Languages Used

The developed framework employs Semantic Web [10] languages, which are XML-based languages, for modelling and providing semantic description of capabilities and status of PaP systems. These descriptions are human-readable and machine comprehensive.
This section will introduce briefly the languages used in the data models, though they are only meant to give the reader enough information to understand the models described in the following sections. For more supplementing information on the languages, please refer to the references indicated in the subsections below.

4.3.1 Common Information Model

Common Information Model (CIM) [11] is a language for describing network management information in standard MOF (Managed Object Format) and XML format. This type of information described could be components like applications, databases, events, network and physical entities. CIM is an information model, a conceptual view of the managed environment where the notions of capabilities and status are represented together as parts of an object's properties. CIM is used in TAPAS for presentation of capability and status information [7]. CIM may be represented using different approaches. This report will represent the CIM instances in both UML graphical notation and XML serialisation. For an introduction to UML the reader is referred to [12].

CIM makes it possible to describe the exact semantics and behaviour of managed entities in a way that is readable for both humans and machines. It is also possible to extend the common classes to include any behaviour specific to the components.

4.3.1.1 XML representation

References [13]

This report represents CIM elements and messages in XML, and therefore will the notation be described briefly in this subsection. Computer and network components are referred to as instances when they are represented using XML. Instances in XML use the tag <INSTANCE>. Instances may be described using properties, property arrays or property references. When we want to represent single CIM properties the tag <PROPERTY> is being used. A property contains a single <VALUE> element describing the value of the property. Property arrays use the tag <PROPERTY.ARRAY> when representing CIM property types that contain array values. Such arrays are represented by the tag <VALUE.ARRAY> and contain value elements. Property references use the tag <PROPERTY.REFERENCE> and are used if one would want to link elements. These references contain value references represented by the tag <VALUE.REFERENCE>. Value references can be further identified by the tag <INSTANCENAME> that is used to define the location of an instance.

For additional information about CIM the reader is referred to [11] and [13].

4.3.2 XML Declarative Description

XML Declarative Description (XDD) employs XML's description format for encoding and exchange of structured data and documents on the Web along with Declarative Description theory for direct representation of data items, encoded in XML-based application languages. XDD aims to add the control to describe axioms, conditions, constraints and similar with a more expressive power. XDD is used in TAPAS to describe play definitions [7], which consists of capability and status requirements of each role in the play as well as the play configuration constrains and reconfiguration rules.

In order to comprehend the descriptions using XDD it is essential to understand the concept of XML clauses and the usage of variables. An XDD description is a set of XML clauses where each XML clause consists of a head and a body. The body contains XML expressions with predefined XML constraints. XML expressions can carry variables and they may be of different types as listed in Table 3.

Variable Type	Prefix	Specialization into
Name (N-variables)	$N	Element types or attribute names
String (S-variables)	$S	Strings
Attribute-value-pairs (P-variables)	$P	Sequences of attribute-value pairs
XML expressions (E-variables)	$E	Sequences of XML expressions
Intermediate expressions (I-variables)	$I	Parts of XML expressions

Table 3: Variables Types

The table defines five disjoint classes of variables with different syntactical usage and specialization characteristics.

Variables are represented with a $ followed by a capital letter telling the type of the variable, then the name is provided. For example $S: deviceID denotes a String-variable named deviceID which is instantiable into only a string, while $E: otherCapabilities is an Expression-variable instantiable into a list of XML expressions representing a sequence of objects or attributes.

The reader is referred to [14] for a more detailed introduction to XML clauses, variable usage and other concepts of the XDD language.

4.3.3 Resource Description Framework

Resource Description Framework (RDF) is a language created to represent metadata, Metadata is information about information. The language describes such data in form of statements where the properties of subjects are represented. RDF is used in TAPAS for representation of messages [1]. RDF may be represented using a graphical approach or by using XML [15]. This report will use XML descriptions, together with UML models similar to graphical RDF models to facilitate the explanation of models. For more basic knowledge in how to effectively use RDF the reader is referred to [15].

5 Intelligent Printing Management System

This project aims to model an Intelligent Printing Management (IPM) system as an application of the developed framework and with the use of PaP. IPM will be demonstrated along with certain application scenarios. The functionality of the application will first be introduced along with the concept, the idea behind IPM, and with its architectural overview and the different roles played. A system description is then proposed. Different data models will be used to describe capability and status descriptions, role specifications and descriptions of configuration and reconfiguration rules. This section ends with a manuscript used to describe the role behaviour of the IPM Manager role.

5.1 Introduction to Intelligent Printing Management system

Resource: [7]

IPM is a concept in managing network and Internet shared printers. The basic assumption is a set of printers serving tens or hundreds of users. Such a system demands a rather high degree of configuration flexibility and must comprise mechanisms such as agreed-on-policies and priority matching. Users of these network printers have to check the availability of printers and their print job queues. In most printer settings there is a high probability of inefficiency in terms of printer usage and achieved tasks. Organizations could actually save in larger amounts of money if they paid a greater deal of attention to what is being printed and where it is being printed, along with a detailed set of policies and rules regarding document sizes, working hours and types of printers. However, queues of print jobs will not be dealt with if there is any congestion at a printer, or any type of operational errors. These jobs could be lost, if someone resets the queue. The jobs could also be double-printed if anyone decides to print them on another printer, in case the print job queue is not accessible and hence print jobs cannot be cancelled.

The following section gives you a typical example of a printing scenario and some reasons why the inefficiency in terms of printer usage is so high. University students are probably one of the target groups that print out most frequently and perhaps too often. Most likely because there are a rather large number of them and they all need to print out documents, foils, articles, papers, lecture notes and home assignments during semester. Most universities do not put a limit on the number of pages each student can print out, which makes students less attentive in order to prevent unnecessary printouts. Generally, students do not bother to print out pages recto-verso and with several foils on each side. They print out new lecture notes and foils that they already have printed out earlier because they forgot them at home, or they do not know where they organized and classified them, or they simply lost them. Most computer halls serve many computers, but only one printer. Students may therefore encounter congestion at a printer during working hours due to large documents that are sent throughout the day. They hang over the printer waiting for their document to emerge, get irritated and send out another copy, or simply leave because they have a class to attend. They do not bother picking up their copy later and chose rather to reprint it out on another occasion. These are some of the reasons of unnecessary printouts and IPM is simply a solution to the problems mentioned above.

5.2 Describing a Play

In order to describe a scenario based on an application, the application itself need first to be described. TAPAS applications come in form of plays, where each play consists of roles. Each role has its behaviour defined by manuscripts. Describing a play therefore involves specifying the behaviour for each role. Applications may also have requirements related to the system environment, in order to make sure that the application will function as intended. It is also important to specify what will happen if requirements are not fulfilled or if errors occur.

5.3 Architectural Overview

Figure 10 illustrates a basic configuration of a network using the IPM application which consists of several different plugged-in printers controlled and managed by a node running IPM manager role (Configuration Manager). The IPM manager role is detectable to every user’s terminal wanting to get access to the different printers.

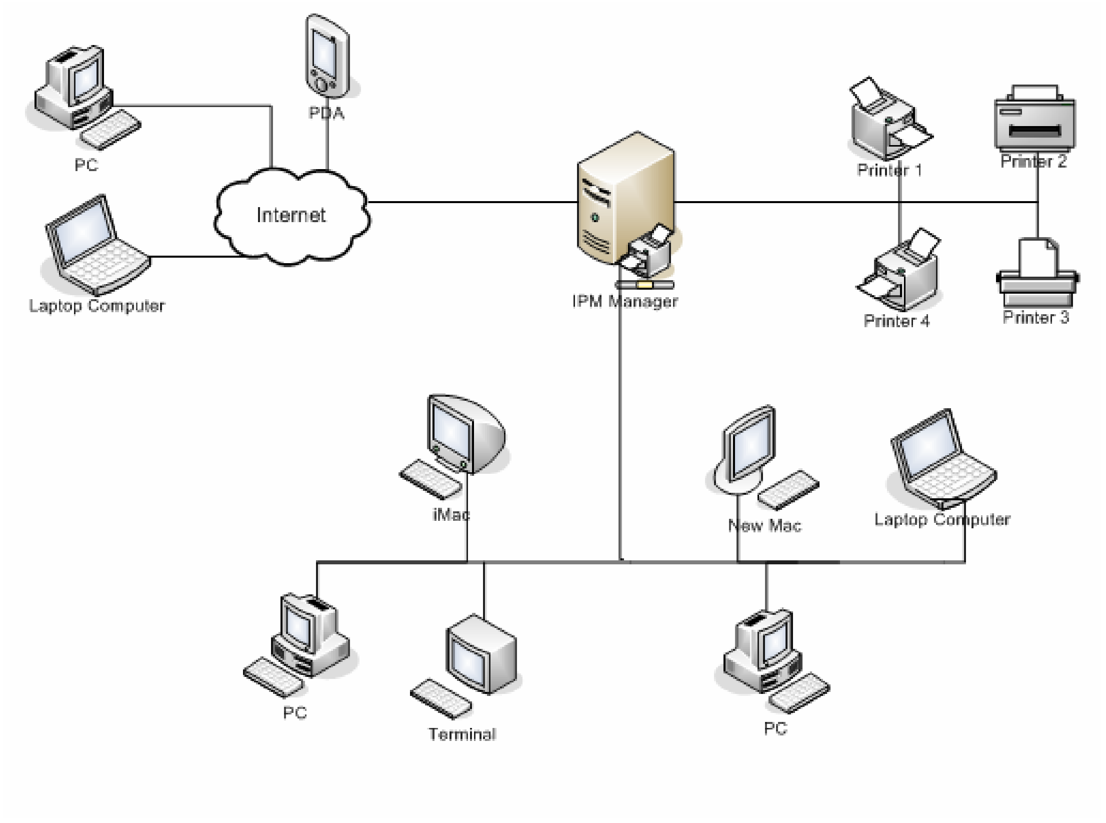


Figure 10: Basic network configuration for IPM

A basic configuration of a network using the IPM application consists of several different plugged-in printers controlled and managed by a node running IPM manager role.

There are different roles that have been defined for the IPM play, and these are shown in Table 4.

Role	**Description**
Document Master (DocMaster)	A print server role for printing black-and-white documents.
Graphic Master (GraphicMaster)	A print server role for handling colour and graphic documents.
Power Point Master (PPMaster)	A print server role for printing four foils per page recto-verso.
Picture Master(PicMaster)	A print server role for printing digital photos and graphics.
IPM Manager (IPMManager)	Responsible for controlling and distributing print jobs to appropriate printer roles, depending on the print job attributes, the current queues of each printer and the job owner's additional information
Print Client (PrintClient)	An application program to which users use for sending print jobs to IPM Manager.

Table 4: IPM Roles

These are roles that take part in the IPM System, where the IPM Manager and the Print Client are mandatory roles, though the print server roles may differ from the one described.

High performance printers can often constitute more than just one print server role, and a print server role can be carried out by one or several physical printers. For instance, a high-speed, laser, colour printer may be configured to play both DocMaster, GraphicMaster and PPMaster roles, while the DocMaster and PPMaster role can be additionally carried out by another black-and-white, laser printer. It is essential to note that in a real application scenario, there could be more varieties and more complicated types of print server roles for which different groups of users have different access control, though these will not be represented here.

During system start up, IPM Manager and server objects will be installed and configured. They will receive their actual behaviour in manuscripts included in a play definition which also consists of capability and status requirement definition. Clients can be plugged in later on nodes that support the TAPAS architecture. From the dynamic configuration point of view, it is important that the play and role requirements are achieved before installing specific roles at specific nodes. Let for instance consider that the IPM manager installation could be required by its role, in a specific play version that it should run on a node with a large disk capacity, a minimum clock speed for the processor and a reliable network connection. Moreover, printer server installations would require some similar options in addition to driver and network address availability.

As explained in section 4.2 the dynamic configuration framework for PaP systems (cf. Figure 9), we perceive that IPM Manager is equivalent to CM. IPM generates a configuration plan at system start up, which gets executed by the Service Installer. Dynamic configuration is handled by IPM Manager by appraisal and adaptation to new changes in the IPM System. In practise, this would be the way how IPM solves out and handles trouble reports when it receives a message. Both the print clients and the print server roles can report relevant troubles to IPM Manager. For each type of trouble report, the IPM Manager acts according to some basic built-in rules or according to the play dynamic reconfiguration rules defined.

5.4 System Description

As depicted in Figure 11, all print jobs are routed to the IPM manager queue where they are buffered up according to the First in First Out (FIFO) queue handling method. A print application consists of print job attributes (header) and the document to print out (body). These attributes are often a list of default settings, although the list may contain more specified adjustments chosen by the print application’s owner for a successful printing. Some are the same as, or extensions of the printer properties that the author found best for printing, and some of these may be embedded in the document depending on the format. In some cases, the embedded information will need to be overridden because it does not make sense for the target printer, e.g. selecting the paper from a certain tray.

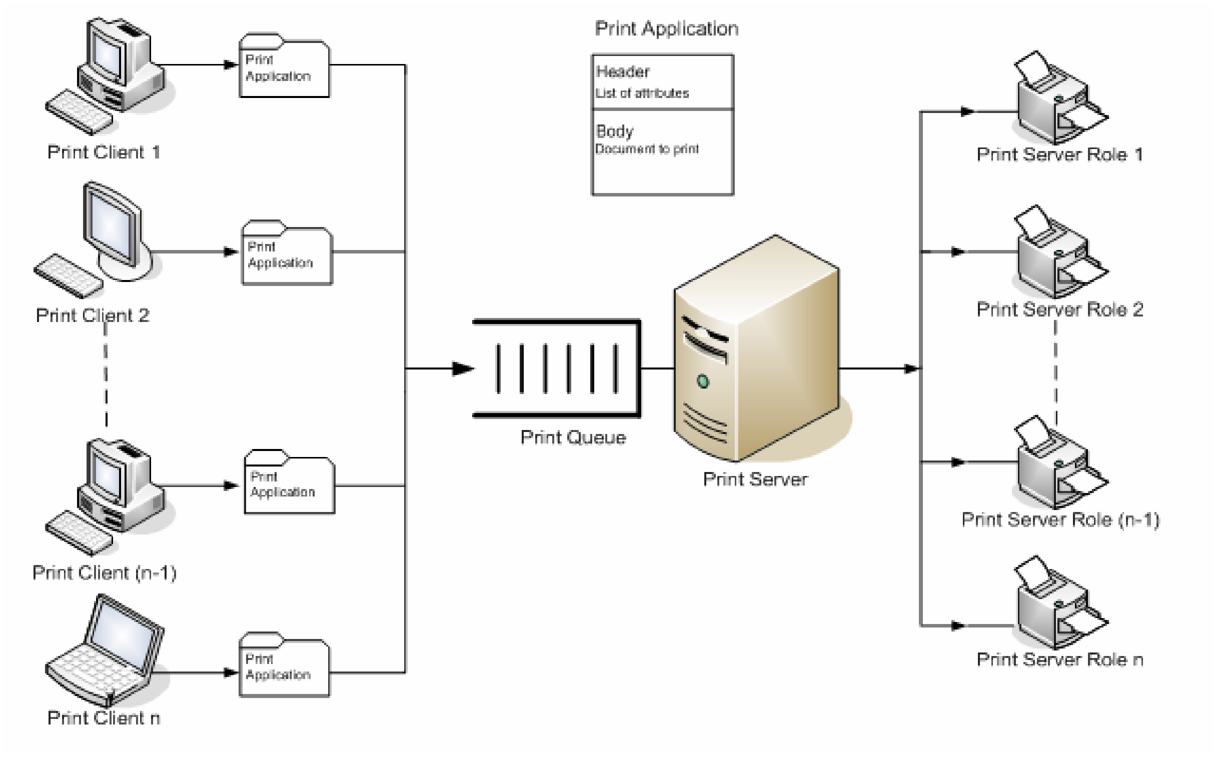


Figure 11: The modelling of the queue

Print applications are sent from Print Clients and buffered up in the Print Queue. The Print Server then analyzes the print job attributes and finds the most appropriate Print Server Role to execute the print job.

The IPM manager finds an appropriate print server role according to the list of attributes that follow with the header. Each print application also contains its unique identification number, in order to differentiate them from each other. If a print server role can match all the requirements of the print application, the IPM Manager sends the print application to be executed at the given printer. IPM Manager also adds the print application to its own print job list, so that it can always supervise which print application has been successfully printed or not. Each time a print job has emerged, the corresponding print server role sends a confirmation message with the print job identification number to the IPM Manager. IPM Manager then deletes the print application from its print job list with the corresponding print job identification number. It also sends a confirmation message to the print client that the print application has been successfully printed with the name of the printer. If there should exist more than one print server role capable of handling the job, the IPM Manager selects the preferred one according to the length of the queue and to the amount of toner left. Each printer has its own FIFO queue, which buffers up the incoming printing jobs where they wait until they get executed.

CSRep contains status and specifications of capabilities offered by the printer nodes in the system. It stores all the different capabilities of each printer, their current status and the print queue length with its print jobs. The CSEMon is responsible for keeping the CSRep updated, and it listens to events indicating changes in the system. CSEMon informs the IPM Manager if it observes changes that will prevent the system from functioning in the most optimal mode, e.g. a node that has insufficient capabilities to execute its functionality. CSEMon checks the status of each printer server role every 10 second by sending a multicast message. Each printer server role replies with a status update message containing a list of attributes and their status which includes the print queue with its print jobs.

After the IPM Manager has sent a print application to be executed at a given printer, it also sends the print job identification along with the estimated printing time and printer name to the CSEMon, which stores this information. The estimated printing time consists of the estimated queue length, which it retrieved from the CSRep, along with the estimated time for the print application it just sent away. The estimated queue length retrieved from CSRep represents the estimated time it would take before all the print jobs in the print queue has been executed. IPM Manager just adds the print application's estimated time to the estimated queue length before sending the new estimated printing time to CSEMon.

CSEMon will now monitor the print applications in their respective queues and starts a timer, which equals the estimated printing time it just received from the IPM Manager. This is necessary in order to detect if a print application has been executed or not. If a print application has been executed before timeout, then it stops the print job's timer. Should a timeout occur before the print job has emerged, then CSEMon inquires the problem by examining the status of the printer in order to find out what the printer's error state is. Additionally CSEMon sends a *Trouble Report* to the IPM Manager indicating the printer's problem.

Upon the receipt of a trouble report, the IPM Manager analyses the problem, fetches related information from the PlayRep and the CSRep, and produces a service reconfiguration plan.

This plan is to be executed by the SR. If the trouble report is due to an error state in a printer, then the print jobs will be moved from the printer's print queue. Each print job will be analysed independent before the IPM Manager inquires the CSRep for a new print server role that matches the print job's attributes. It then finds the most appropriate printer for the print application and sends an announcement message to the Print Client that the print job has been redirected to another printer. The Print Client Identification which was previously stored in IPM Manager's print job list will now be updated with the print job's new printer. Table 5 shows that each print application in the print job list is associated with its print job identification, print client identification and its printer name. The printer name tells which printer will execute the print job. The print job list is compulsory so that the IPM Manager can recover lost print applications.

Print Job Identification	Print Application	Print Client Identification	Printer Name

Table 5: Print Job List

Each print application in the print job list is associated with its print job identification, print client identification and its printer name.

5.5 IPM System Data Models

A basic configuration of a network using the IPM application consists of roles played by different actors. This section will describe the role specification with a model already provided from [7] and introduce new ones. Different data models will be used to describe capability and status descriptions, role specifications and descriptions of configuration and reconfiguration rules.

5.5.1 Document Master (DocMaster)

The actor who is playing the role DocMaster needs to be a print server role and have the ability to print black-and-white documents. Figure 12 describes an XML clause, a model suggested from [7] to represent role specifications, with the capabilities and status requirements of the print server role DocMaster. The figure shows simple examples of modelling both the capability and status requirements with both graphical (using UML class diagram notations) and textual presentation of the clause. It can be read as follows:

(A) Head of clause

An actor playing the role DocMaster can be installed into $S:nodeX, and $S:nodeX being an instance of the class CIM_Printer.

If the body of the clause

(B)

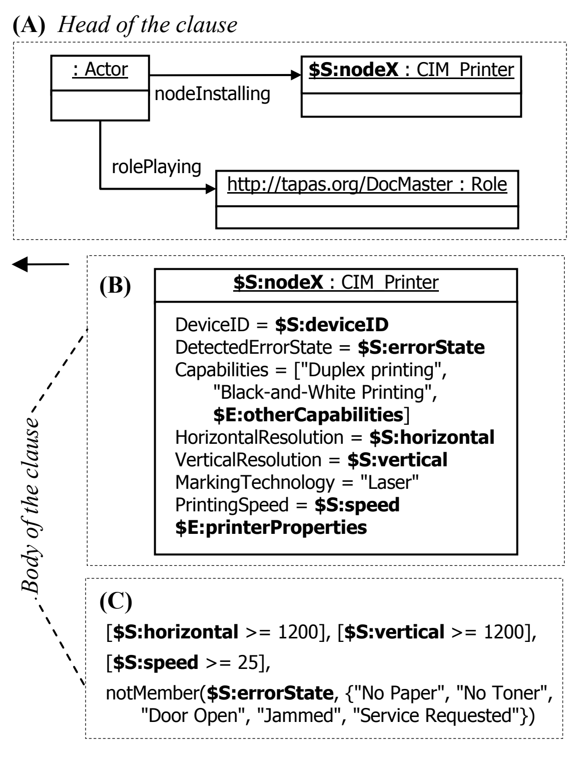
$S:nodeX is not occupied and can proffer

- Duplex printing and black-and-white capabilities, and
- Laser marking technology

(C)

Where the following supplementary conditions on $S:nodeX's capabilities and status need to be satisfied

- [$S:horizontal >= 1200] and [$S:vertical >= 1200]: the horizontal and the vertical resolutions of the print function are at least 1200 dots per inch (dpi),
- [$S:speed] >= 25]: the printing speed must be greater than 25 ppm, and
- notMember($SerrorState, {"No Paper", "No Toner", "Door Open", "Jammed", "Service Requested"}): none of the given error states in the list is detected.


(A) Head of the clause
: Actor
nodeInstalling
$S:nodeX : CIM_Printer
rolePlaying
http://tapas.org/DocMaster : Role
(B)
$S:nodeX : CIM_Printer
DeviceID = $S:deviceID
DetectedErrorState = $S:errorState
Capabilities = ["Duplex printing", "Black-and-White Printing", $E:otherCapabilities]
HorizontalResolution = $S:horizontal
VerticalResolution = $S:vertical
MarkingTechnology = "Laser"
PrintingSpeed = $S:speed
$E:printerProperties
Body of the clause
(C)
[$S:horizontal >= 1200], [$S:vertical >= 1200],
[$S:speed >= 25],
notMember($S:errorState, {"No Paper", "No Toner", "Door Open", "Jammed", "Service Requested"})

(a) Graphical notation.

```
<Actor>
  <rolePlaying  rdf:resource="http://tapas.org/DocMaster"/>
  <nodeInstalling  rdf:resource=$S:nodeX />
</Actor>
   ←  <INSTANCE  ClassName="CIM_Printer">
          <PROPERTY  NAME="DeviceID">
              <VALUE>$S:deviceID</VALUE>
          </PROPERTY>
          <PROPERTY  NAME="DetectedErrorState">
              <VALUE>$S:errorState</VALUE>
          </PROPERTY>
          <PROPERTY.ARRAY  NAME="Capabilities">
              <VALUE.ARRAY>
                  <VALUE>Duplex Printing</VALUE>
                  <VALUE>Black-and-White Printing</VALUE>
                  $E:otherCapabilities
              </VALUE.ARRAY>
          </PROPERTY.ARRAY>
          <PROPERTY  NAME="HorizontalResolution">
              <VALUE>$S:horizontal</VALUE>
          </PROPERTY>
          <PROPERTY  NAME="VerticalResolution">
              <VALUE>$S:vertical</VALUE>
          </PROPERTY>
          <PROPERTY  NAME="MarkingTechnology">
              <VALUE>Laser</VALUE>
         </PROPERTY>
          <PROPERTY  NAME="PrintingSpeed">
              <VALUE>$S:speed</VALUE>
          </PROPERTY>
          $E:printerProperties
      </INSTANCE>,
      [$S:horizontal >= 1200],  [$S:vertical >= 1200],
      [$S:speed >= 25],
      notMember($S:errorState, {"No Paper", "No Toner", "Door
      Open", "Jammed", "Service Requested"})
```

(b) XML serialisation.

Figure 12: XDD description of capability & status requirements of the role Document Master [7]

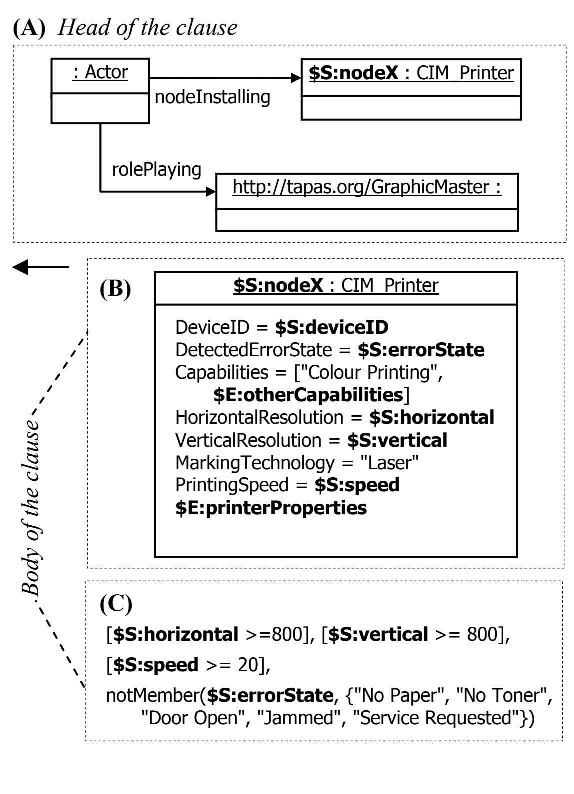
The head of the XML clause shows an actor playing the role DocMaster that is installed on a CIM_printer called nodeX. The body declare the conditions that must be accomplished before nodeX can install an actor playing the role DocMaster [7].

The header depicts an actor playing the role DocMaster, which is installed on a CIM_Printer called nodeX. XDD is used in this data model to state that there are some requirements that must be fulfilled before an actor might state to play a given role. If CIM_Printer shall be able to install the role, then there are some conditions that must be fulfilled. These are shown in the body of the clause. NodeX must be of type CIM_Printer and offer several capabilities. It must have the capability of duplex printing and black-and-white printing, have a horizontal resolution greater than 1200, a vertical resolution greater than 1200 and it must also offer a printing speed greater than 25 ppm. Besides this, it also need to offer laser as marking techonology, and the node can not be in any of the following error states: no paper, no toner, door open, jammed or service requested.

5.5.2 Graphic Master (GraphicMaster)

The actor who is playing the role Graphic Master needs to be a print server role and have the ability to print colour and graphic documents. Figure 13 describes an XML clause, an example of modelling proposed by [7], with the capabilities and status requirements of the print server role Graphic Master. The figure shows both graphical (using UML class diagram notations) and textual presentation.

Like the XDD description for the Document Master (5.5.1), this clause Graphic Master defines that an actor playing the role Graphic Master can be installed into $S:nodeX if it fulfils the following requirements: $S:nodeX being an instance of the class CIM_Printer, and offers laser colour printing capability with the resolution 800*800 dpi, and be able to print at least 20 ppm. Its status must not be one of "No Paper", "No Toner", "Door Open", "Jammed" and "Service Requested.


(A) Head of the clause
: Actor
nodeInstalling
$S:nodeX : CIM_Printer
rolePlaying
http://tapas.org/GraphicMaster :
(B)
$S:nodeX : CIM_Printer
DeviceID = $S:deviceID
DetectedErrorState = $S:errorState
Capabilities = ["Colour Printing", $E:otherCapabilities]
HorizontalResolution = $S:horizontal
VerticalResolution = $S:vertical
MarkingTechnology = "Laser"
PrintingSpeed = $S:speed
$E:printerProperties
Body of the clause
(C)
[$S:horizontal >=800], [$S:vertical >= 800],
[$S:speed >= 20],
notMember($S:errorState, {"No Paper", "No Toner", "Door Open", "Jammed", "Service Requested"})

(a) Graphical notation.

```
<Actor>
  <rolePlaying  rdf:resource="http://tapas.org/GraphicMaster"/>
  <nodeInstalling  rdf:resource=$S:nodeX />
</Actor>
  ←  <INSTANCE  ClassName="CIM_Printer">
         <PROPERTY  NAME="DeviceID">
             <VALUE>$S:deviceID</VALUE>
         </PROPERTY>
         <PROPERTY  NAME="DetectedErrorState">
             <VALUE>$S:errorState</VALUE>
         </PROPERTY>
         <PROPERTY.ARRAY  NAME="Capabilities">
             <VALUE.ARRAY>
                 <VALUE>Colour Printing</VALUE>
                 $E:otherCapabilities
             </VALUE.ARRAY>
         </PROPERTY.ARRAY>
         <PROPERTY  NAME="HorizontalResolution">
             <VALUE>$S:horizontal</VALUE>
         </PROPERTY>
         <PROPERTY  NAME="VerticalResolution">
             <VALUE>$S:vertical</VALUE>
         </PROPERTY>
         <PROPERTY  NAME="MarkingTechnology">
             <VALUE>Laser</VALUE>
         </PROPERTY>
         <PROPERTY  NAME="PrintingSpeed">
             <VALUE>$S:speed</VALUE>
         </PROPERTY>
         $E:printerProperties
     </INSTANCE>,
     [$S:horizontal >= 800],  [$S:vertical >= 800],
     [$S:speed >= 20],
     notMember($S:errorState, {"No Paper", "No Toner", "Door
     Open", "Jammed", "Service Requested"})
```

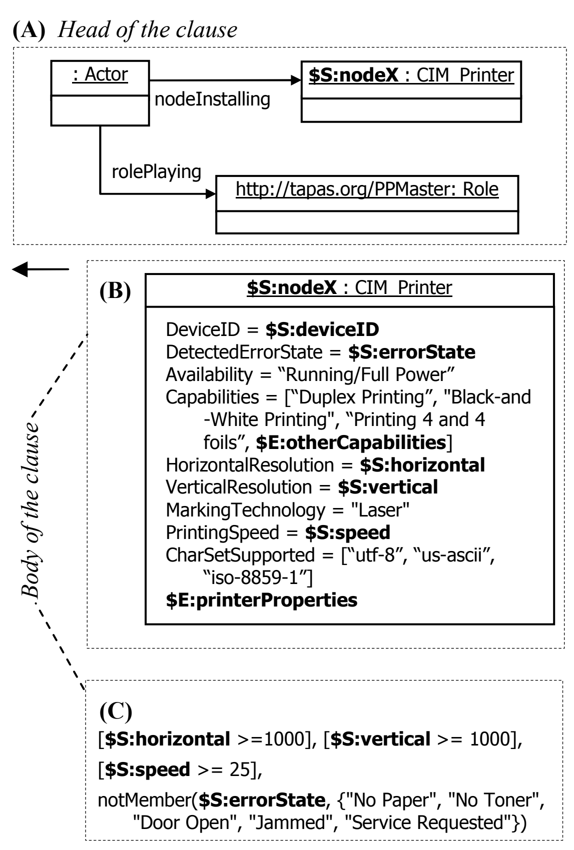
(b) XML serialisation.

Figure 13: XDD description of capability & status requirements of the role Graphic Master [7]

5.5.3 Power Point Master (PPMaster)

Power Point Master is a print server role for printing four foils per page recto-verso. This print server role is proposed in attempt to reduce the number unnecessary printouts, since most organizations and institutions do not put a limit on the number of pages e.g. each employee or student can print out. Furthermore, print clients are not attentive enough in order to prevent unnecessary printouts, and they often do not bother to print out pages recto-verso and with several foils on each side. The Power Point Master is simply a solution to the problem mentioned above.

Figure 14 describes an XML clause, with the capabilities and status requirements of the print server role Power Point Master. The figure shows both graphical (using UML class diagram notations) and textual presentation. The requirements listed are only examples of how the role Power Point Master could be. The complete representation of the XML textual description can be found in Appendix B.


(A) Head of the clause
: Actor
nodeInstalling
$S:nodeX : CIM_Printer
rolePlaying
http://tapas.org/PPMaster: Role
(B)
$S:nodeX : CIM_Printer
DeviceID = $S:deviceID
DetectedErrorState = $S:errorState
Availability = "Running/Full Power"
Capabilities = ["Duplex Printing", "Black-and -White Printing", "Printing 4 and 4 foils", $E:otherCapabilities]
HorizontalResolution = $S:horizontal
VerticalResolution = $S:vertical
MarkingTechnology = "Laser"
PrintingSpeed = $S:speed
CharSetSupported = ["utf-8", "us-ascii", "iso-8859-1"]
$E:printerProperties
Body of the clause
(C)
[$S:horizontal >=1000], [$S:vertical >= 1000],
[$S:speed >= 25],
notMember($S:errorState, {"No Paper", "No Toner", "Door Open", "Jammed", "Service Requested"})

(a) Graphical notation.

```
<Actor>
  <rolePlaying  rdf:resource="http://tapas.org/PPMaster"/>
  <nodeInstalling  rdf:resource=$S:nodeX />
</Actor>
  ←   <INSTANCE  ClassName="CIM_Printer">
          <PROPERTY  NAME="DeviceID">
               <VALUE>$S:deviceID</VALUE>
          </PROPERTY>
          <PROPERTY  NAME="DetectedErrorState">
               <VALUE>$S:errorState</VALUE>
          </PROPERTY>
         <PROPERTY  NAME="Availability">
               <VALUE>Running/Full Power</VALUE>
          </PROPERTY>
          <PROPERTY.ARRAY  NAME="Capabilities">
               <VALUE.ARRAY>
                    <VALUE>Duplex Printing</VALUE>
                   <VALUE>Black-and-White Printing</VALUE>
                   <VALUE>Printing 4 and 4 foils</VALUE>
                    $E:otherCapabilities
               </VALUE.ARRAY>
          </PROPERTY.ARRAY>
          <PROPERTY  NAME="HorizontalResolution">
               <VALUE>$S:horizontal</VALUE>
          </PROPERTY>
          <PROPERTY  NAME="VerticalResolution">
               <VALUE>$S:vertical</VALUE>
          </PROPERTY>
          <PROPERTY  NAME="MarkingTechnology">
               <VALUE>Laser</VALUE>
         </PROPERTY>
          <PROPERTY  NAME="PrintingSpeed">
               <VALUE>$S:speed</VALUE>
          </PROPERTY>
         <PROPERTY.ARRAY  NAME="CharSetSupported">
               <VALUE.ARRAY>
                    <VALUE>utf-8</VALUE>
                   <VALUE> us-ascii </VALUE>
                   <VALUE> iso-8859-1</VALUE>
               </VALUE.ARRAY>
          </PROPERTY.ARRAY>
          $E:printerProperties
      </INSTANCE>,
      [$S:horizontal >= 1000],  [$S:vertical >= 1000],
      [$S:speed >= 25],
      notMember($S:errorState, {"No Paper", "No Toner", "Door
      Open", "Jammed", "Service Requested"})
```

(b) XML serialisation.

Figure 14: XDD description of capability & status requirements of the role Power Point Master

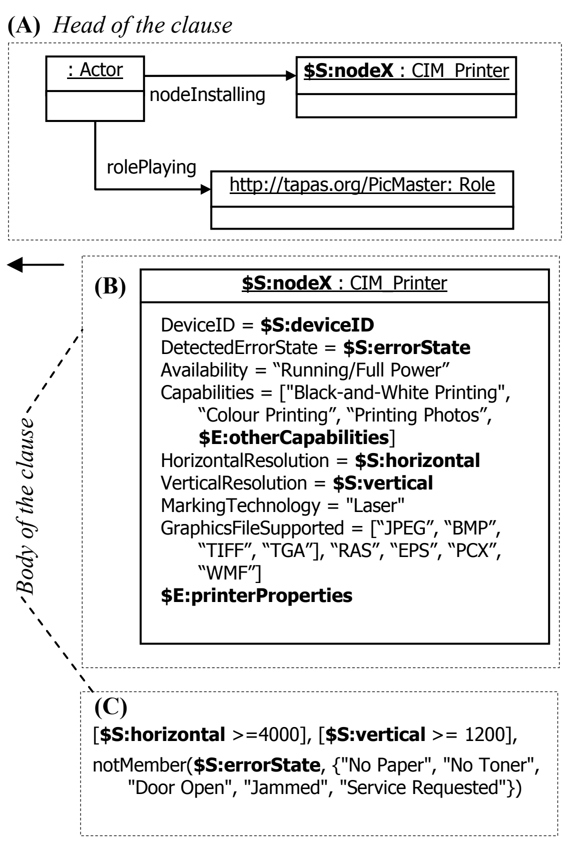
The actor who is playing the Power Point Master role needs to be a print server role and have the ability to print documents containing foils with some default settings. These settings are set to always print out documents containing foils to four foils per page recto-verso. When the IPM Manager receives a document containing only foils, then it automatically sends the print job to a printer that is playing the Power Point Master role to execute the print job.

Taking a close look at Figure 14, we see that the clause for the Power Point Master defines an actor playing the role Power Point Master and installable into $S:nodeX if it fulfils the subsequent constraints: S:nodeX being an instance of the class CIM_Printer, and offers laser black-and-white printing capability with the resolution 1000*1000 dpi, and be able to print at least 25 ppm. Its status must not be one of "No Paper", "No Toner", "Door Open", "Jammed" and "Service Requested, and character set chosen must be one of "utf-8", "us-ascii" and "iso-8859-1".

5.5.4 Picture Master (PicMaster)

Picture Master is a print server role for printing digital photos and graphics. This print server role shall make it easy to print studio-quality colour and black-and-white photos in different formats and sizes. It switches easily from 4 x 6-inch to letter-size printing, since printers can provide built-in paper trays. Photos can be printed with or without borders on standard sizes of photo papers.

Figure 15 defines the capability and status requirements of the Picture Master. After studying the previous print server roles, the reader should have knowledge enough to understand how role specifications may be specified using the proposed XML based data model. Picture Master specifications will therefore not be described more in detail. The complete representation of the XML textual description can be found in Appendix B.


(A) Head of the clause
: Actor
nodeInstalling
$S:nodeX : CIM_Printer
rolePlaying
http://tapas.org/PicMaster: Role
(B)
$S:nodeX : CIM_Printer
DeviceID = $S:deviceID
DetectedErrorState = $S:errorState
Availability = "Running/Full Power"
Capabilities = ["Black-and-White Printing", "Colour Printing", "Printing Photos", $E:otherCapabilities]
HorizontalResolution = $S:horizontal
VerticalResolution = $S:vertical
MarkingTechnology = "Laser"
GraphicsFileSupported = ["JPEG", "BMP", "TIFF", "TGA"], "RAS", "EPS", "PCX", "WMF"]
$E:printerProperties
Body of the clause
(C)
[$S:horizontal >=4000], [$S:vertical >= 1200],
notMember($S:errorState, {"No Paper", "No Toner", "Door Open", "Jammed", "Service Requested"})

(a) Graphical notation.

```
<Actor>
  <rolePlaying  rdf:resource="http://tapas.org/PicMaster"/>
  <nodeInstalling  rdf:resource=$S:nodeX />
</Actor>
  ←   <INSTANCE  ClassName="CIM_Printer">
          <PROPERTY  NAME="DeviceID">
               <VALUE>$S:deviceID</VALUE>
          </PROPERTY>
          <PROPERTY  NAME="DetectedErrorState">
               <VALUE>$S:errorState</VALUE>
          </PROPERTY>
         <PROPERTY  NAME="Availability">
               <VALUE>Running/Full Power</VALUE>
          </PROPERTY>
          <PROPERTY.ARRAY  NAME="Capabilities">
               <VALUE.ARRAY>
                    <VALUE>Black-and-White Printing</VALUE>
                    <VALUE>Colour Printing</VALUE>
                    <VALUE>Printing Photos</VALUE>
                     $E:otherCapabilities
               </VALUE.ARRAY>
          </PROPERTY.ARRAY>
          <PROPERTY  NAME="HorizontalResolution">
               <VALUE>$S:horizontal</VALUE>
          </PROPERTY>
          <PROPERTY  NAME="VerticalResolution">
               <VALUE>$S:vertical</VALUE>
          </PROPERTY>
          <PROPERTY  NAME="MarkingTechnology">
               <VALUE>Laser</VALUE>
          </PROPERTY>
          <PROPERTY.ARRAY  NAME="GraphicsFileSupported">
               <VALUE.ARRAY>
                     <VALUE>GIF</VALUE>
                      <VALUE>JPEG</VALUE>
                     <VALUE>BMP</VALUE>
                     <VALUE>TIFF</VALUE>
                     <VALUE>TGA</VALUE>
                     <VALUE>RAS</VALUE>
                     <VALUE>EPS</VALUE>
                     <VALUE>PCX</VALUE>
                      <VALUE>WMF</VALUE>
                     <VALUE>PNG</VALUE>
               </VALUE.ARRAY>
          </PROPERTY.ARRAY>
          $E:printerProperties
      </INSTANCE>,
      [$S:horizontal >= 4000],  [$S:vertical >= 1200],
      notMember($S:errorState, {"No Paper", "No Toner", "Door
      Open", "Jammed", "Service Requested"})
```

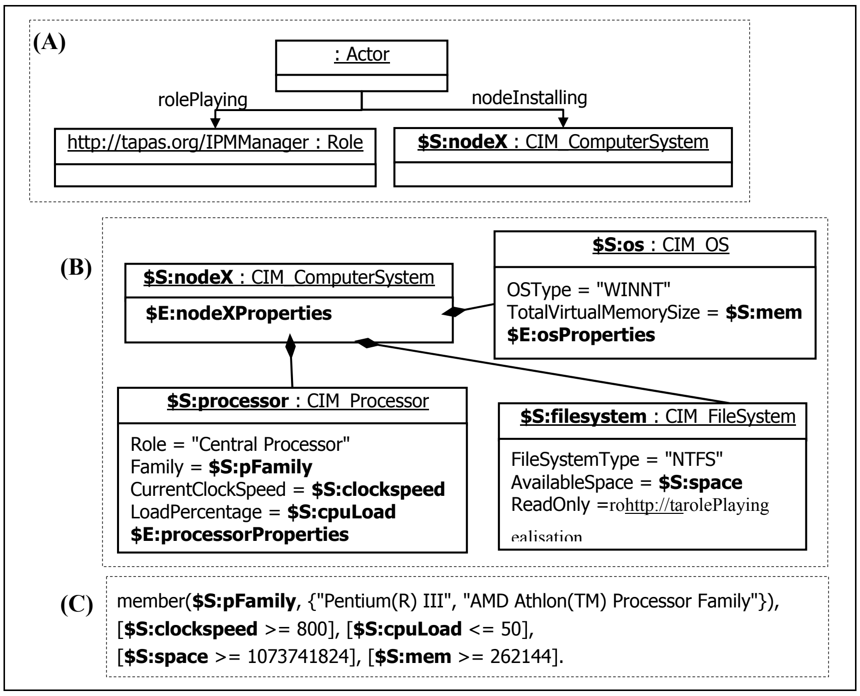
(b) XML serialisation.

Figure 15: XDD description of capability & status requirements of the role Picture Master

5.5.5 IPM Manager

IPM Manager is responsible for controlling and distributing print jobs to appropriate printer roles, depending on the print job attributes, the current queues of each printer and the job owner's additional information. It finds an appropriate print server role which is capable of handling the job. A preferred one will be selected if there should exist more than one print server for the print job.

Figure 16 gives a simple example of modelling the capability and status requirements of the IPM Manager role proposed from [7]. The figure uses only the graphical presentation of the XML clause, while the textual presentation is omitted since it is less illustrative and user-friendly than the graphical one.



(A)
: Actor
rolePlaying
nodeInstalling
http://tapas.org/IPMManager : Role
$S:nodeX : CIM_ComputerSystem
(B)
$S:nodeX : CIM_ComputerSystem
$E:nodeXProperties
$S:os : CIM_OS
OSType = "WINNT"
TotalVirtualMemorySize = $S:mem
$E:osProperties
$S:processor : CIM_Processor
Role = "Central Processor"
Family = $S:pFamily
CurrentClockSpeed = $S:clockspeed
LoadPercentage = $S:cpuLoad
$E:processorProperties
$S:filesystem : CIM_FileSystem
FileSystemType = "NTFS"
AvailableSpace = $S:space
ReadOnly =rohttp://tarolePlaying
ealisation
(C)
member($S:pFamily, {"Pentium(R) III", "AMD Athlon(TM) Processor Family"}),
[$S:clockspeed >= 800], [$S:cpuLoad <= 50],
[$S:space >= 1073741824], [$S:mem >= 262144].

Figure 16: XDD description of capability & status requirements of the role IPM Manager [7]

The XDD description of the IPM Manager clause specifies that an actor playing the role IPM Manager can be installed into $S:nodeX if it fulfils the following requirements (B) and (C): $S:nodeX being an instance of the class CIM_ComputerSystem requires that its central processor is either Pentium III or AMD Athlon families with the minimum clock speed of 800 MHz and the load percentage less than 50. It also requires WINNT as installed operating system with the minimum RAM of 262144 KB (256 MB), and the computer's hosted file system must be NTFS with at least the available space of 1073741824 bytes (1 GB).

5.5.6 Configuration Rules

A configuration rule is the plan dynamically generated by the IPM Manager upon the receipt of a service request or a service component request. Figure 17 shows that the data model proposed for describing the configuration rules also makes use of XDD features. The XDD description shown in the figure is an example of a configuration rule defining specified constraints. These constraints may define the combination of roles needed to be part of the play in order to obtain an executable configuration.

One may also specify what roles are to participate, how many actors should play the different roles, and what capabilities should be preferred when there exist more than one actor able to play the role. The XML example in Figure 17 defines a configuration rule of the play http://PaP.org/IPM_1.0. The clause's head (A) states that five roles need to be realised in order to perform the play. Each role is specified by a roleRealisation association. The first association indicates that there must exist exactly one actor constituting the role IPMManager at a node $S:IPM_Node. The other roleRealisation associations specify that the configuration also includes installation of actors realising the following roles: DocMaster, GraphicMaster, PPMaster and PicMaster.

The clause's body, comprising (B) to (G), defines the conditions of the actors to install and the roles to play. Part (B) indicates that reception of a ServiceRequest telling to install the play will trigger the configuration. Part (C) ensures that the $S:IPM_node has satisfactory capabilities and status to install an actor for executing the IPMManager role, like the clause given in section 5.5.5. Part (D) specifies that there are specific requirements to be met by the nodes willing to install the $E:DocMaster role. The nodes' capabilities and status can be equal to those represented in section 5.5.1. Correspondingly, part (E), (F) and (G) specify that $E:GraphicMaster, $E:PPMaster and $E:PicMaster respectively represent a set of nodes which are capable of playing the different roles.

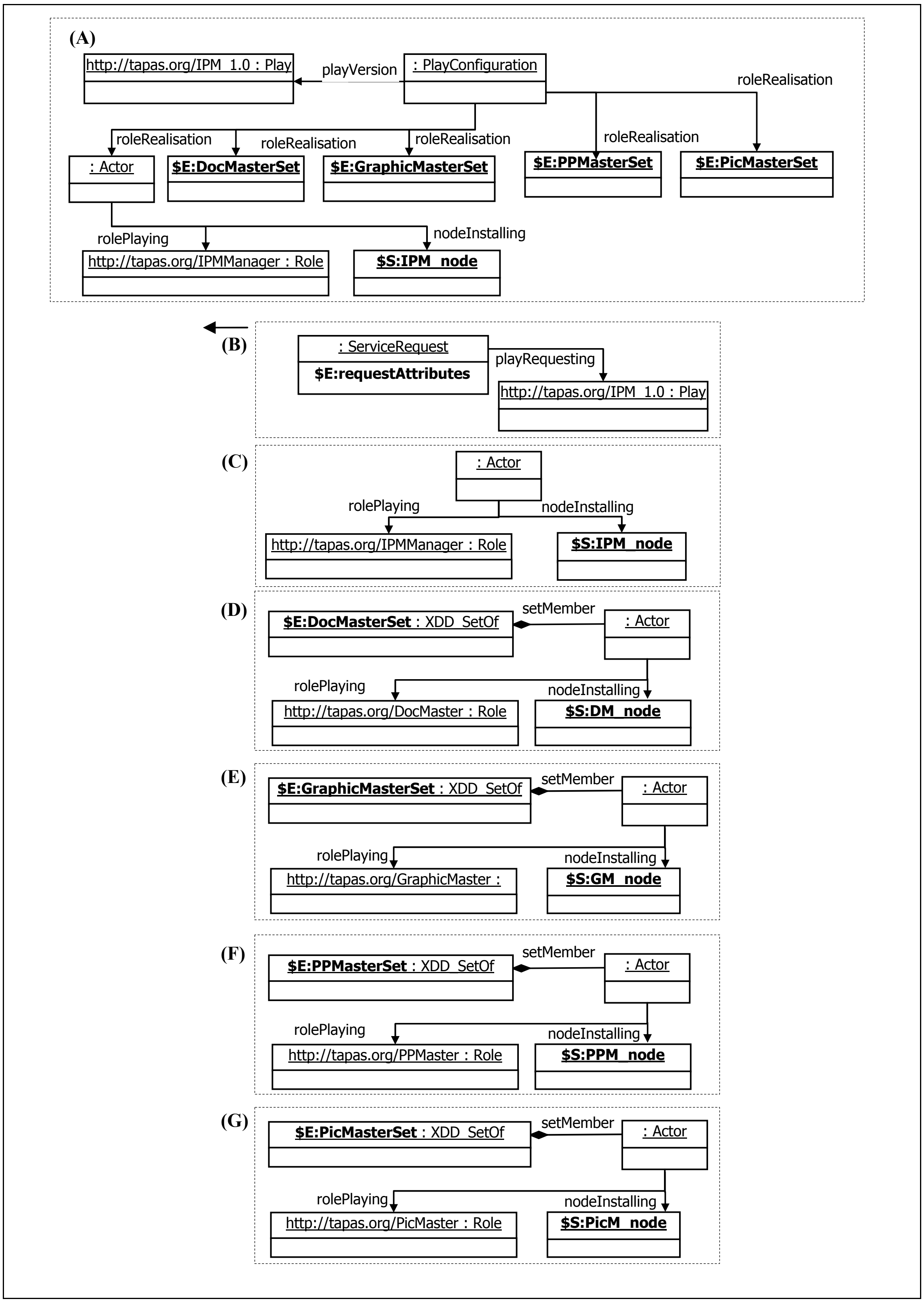


Figure 17: XDD description of a play configuration rule used in the IPM service system

Figure modified from [7]

5.5.7 Reconfiguration Rules

Reconfiguration rules specify actions that should be taken upon receipt of trouble reports. The rules are formalised as XML clauses. The head of each XML clause describes the reconfiguration action to be executed, while its body defines the types, conditions and details to be performed. The IPM Manager reacts when receiving trouble reports by generating an apposite reconfiguration plan according to the defined rule. If there is not a reconfiguration rule specially defined for handling the given trouble, then the IPM Manager performs the default reconfiguration, i.e. the actors involved in the problem will be relocated. Table 6 defines the possible types of reconfiguration.

Reconfiguration Action	Description
No Action	The system performs no action. It disregards the trouble report.
Actor Initialisation	Instantiation of new actor at a specified node, followed by installation of the manuscript defining the actor's behaviour, execution of the actor.
Actor Termination	The specific actor will be terminated and the resources allocated to and consumed by the actor will be released.
Actor Re-initialisation	The specific actor will be terminated and re-initialised at the same node.
Actor Relocation	An actor is moved to another node
Play Reconfiguration	The whole play will be reconfigured, which includes that the best node for executing each actor will be re-determinated and the actor will be relocated to that new location.
Default	Relocation of all the actors involves in the problem.

Table 6: Reconfiguration Actions

Figure 18 shows the XML clause for a specific reconfiguration rule. This clause handles the reconfiguration rule InsufficientCapabilityReport of an actor playing the role IPMManager. The clause's head (A) defines an ActorRelocation plan, which specifies that the actor $S:actorA is to be relocated to $S:newNode under certain conditions; if it has been identified that $S:actorA has insufficient capabilities to continue playing the IPMManager role at node $S:node, and that there exist a node in the system more suitable for playing the IPMManger role. This node is denoted $S:newNode.

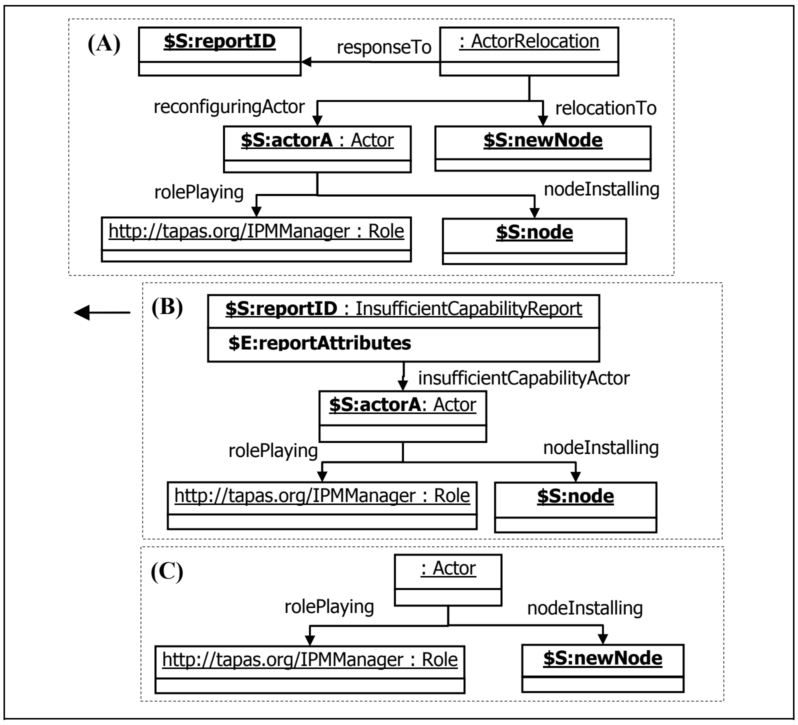


Figure 18: XDD description of a dynamic reconfiguration rule used in the IPM service system [7]

5.6 Internet Printing

Source: [8], [9]

Internet Printing is the process of sending a print job request directly to a printer via the Internet. Imagine when you need to send your Singapore suppliers a report created by your custom application, you can print it directly on one of their printer, without worrying about compatibility and no more expensive fax time or postage. Internet Printing is possible due to the Internet Printing Protocol (IPP), an application level protocol that can be used for distributed printing using Internet tools and technologies and that covers most common situations for printing on the Internet. As depicted in Figure 10, IPM System supports Internet Printing with the use of IPP. IPP is a set of communication rules built on top of the HTTP Internet Standard. The ability to poll a printer and ask about its availability and properties is one of the key features. To function, an IPP printer requires a Universal Resource Identifier (URI), which is typically a Uniform Resource Locator (URL), including an IP address. Since it is built on HTTP, printer may also be configured to require a user name and password. Support for IPP includes leading printer vendors such as Hewlett Packard, Xerox and IBM. These printer vendors, and many others, are incorporating IPP capability into their new printers. There is also a range of hardware and software products that allow any printer to be exposed to the Internet using IPP. Network print servers from companies such as Novell, IBM and Microsoft have IPP features built in.

IPP defines basic handshaking and communication methods and allows the following functions:

- Users may find out about a printer's capabilities
- Users can submit print jobs directly to a printer
- Users can find out the status of a printer or a print job
- Users can cancel a previously submitted job

One of the methods of Internet Printing involves third party or service bureau printing. Many users may not have available the printer that is best for the job, e.g. they have an ink-jet printer, but desire to use a laser colour printer, or perhaps they would like to print a large quantity of a document. In this situation, a user may elect to send a file to a copy or printing centre for reproduction. This is one reason why Internet Printing in the IPM System will be particularly popular with students, small businesses, and business travellers. Interestingly, these requirements will be almost identical to intranet printing situations where a centralized print or copy room is available.

As for the case of Internet Printing in the IPM System, print clients are allowed to choose the printer they wish to use. Their print job will pass trough the IPM Manager, which sends the print job to the requested printer. Should a more appropriate printer be available, then the print clients will get a proposal asking them if they want to change printer. Print clients can deny proposals, since they might want to use the printer nearest the person who the document is addressed to, so that the recipient does not have to wonder which printer executed the print job.

5.7 Print Jobs

There exist three different types of print jobs that the IPM Manager may process. Print Clients may use Internet Printing as described in the previous section, or they can use a specific printer. The last option is to send a print job and let the IPM Manager find the most appropriate printer. These three signals are called InternetPrinting, PrintJobSpec and PrintJobUnspec respectively, and are shown in Figure 19 with their corresponding parameters.

The figure describes one of the states using SDL (for an introduction on SDL the reader is referred to [18]). From the figure we can see that the state Idle handles the three print signals just mentioned above. Moreover, the state Idle receives other signals than those three depicted in the figure. They are not imperative for this example.

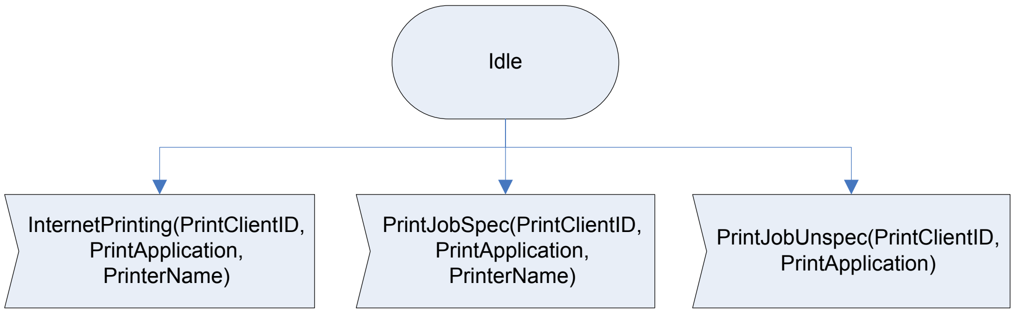

Idle
InternetPrinting(PrintClientID, PrintApplication, PrinterName)
PrintJobSpec(PrintClientID, PrintApplication, PrinterName)
PrintJobUnspec(PrintClientID, PrintApplication)

Figure 19: Print jobs in IPM

The SDL figure shows the three different types of printing that Print Clients may use with their corresponding parameters.

If we take a closer look at the signal InternetPrinting, we see that it contains the following parameters:

- PrintClientID is in this case the IP address of the print client's workstation
- PrintApplication consists of print job attributes (header) and the document to print out (body). These attributes constitute a list of settings chosen by the print application's owner.
- PrinterName is the name that identifies the printer.

5.8 Manuscripts

Manuscripts are the data elements used to describe role behaviour. In IPM different roles are needed to run the IPM System, roles as IPMManager, PrintClient and the different print server roles DocMaster, GraphicMaster, PicMaster and PPMaster. This section describes parts of the manuscripts for the role IPMManager. The full representation of this manuscript can be found in Appendix A. The manuscript of the role IPM Manager contains many different states.

Figure 20 describes one of the states using SDL, but not all the signals are shown. The following section will concentrate on the handling of the signal InternetPrinting.

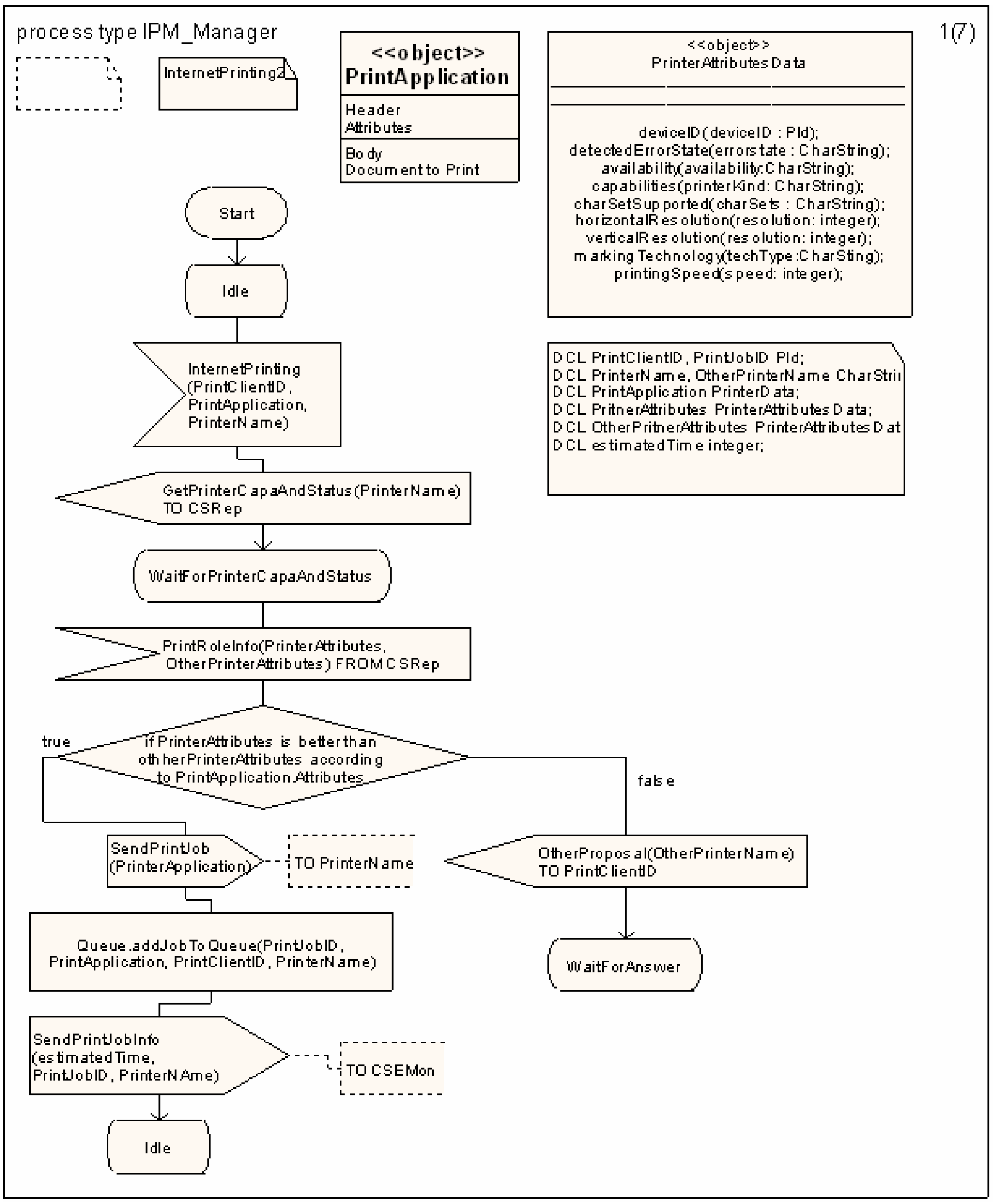

process type IPM_Manager
1(7)
InternetPrinting2
<<object>>
PrintApplication
Header
Attributes
Body
Document to Print
<<object>>
PrinterAttributes Data
deviceID(deviceID : PId);
detectedErrorState(errorstate : CharString);
availability(availability:CharString);
capabilities(printerKind: CharString);
charSetSupported(charSets : CharString);
horizontalResolution(resolution: integer);
verticalResolution(resolution: integer);
markingTechnology(techType:CharSting);
printingSpeed(speed: integer);
DCL PrintClientID, PrintJobID PId;
DCL PrinterName, OtherPrinterName CharStri
DCL PrintApplication PrinterData;
DCL PritnerAttributes PrinterAttributesData;
DCL OtherPritnerAttributes PrinterAttributesDat
DCL estimatedTime integer;
Start
Idle
InternetPrinting (PrintClientID, PrintApplication, PrinterName)
GetPrinterCapaAndStatus(PrinterName) TO CSRep
WaitForPrinterCapaAndStatus
PrintRoleInfo(PrinterAttributes, OtherPrinterAttributes) FROM CSRep
true
if PrinterAttributes is better than othherPrinterAttributes according to PrintApplication.Attributes
false
SendPrintJob (PrinterApplication)
TO PrinterName
OtherProposal(OtherPrinterName) TO PrintClientID
Queue.addJobToQueue(PrintJobID, PrintApplication, PrintClientID, PrinterName)
WaitForAnswer
SendPrintJobInfo (estimatedTime, PrintJobID, PrinterNAme)
TO CSEMon
Idle

Figure 20: Parts of the manuscript of the IPM Manager using SDL

The figure shows the process graph and its transitions specified for the state Idle. What decisions to make depends on the capabilities and status offered by the different print server roles.

From these descriptions the first action taken upon reception of the signal InternetPrinting is fetching information from the CSRep. This is done by sending the signal GetPrinterCapaAndStatus with the PrinterName as parameter. The IPM Manager is then said to enter the state WaitForPrinterCapaAndStatus where it waits for the answer PrintRoleInfo, from the CSRep. The signal PrintRoleInfo contains the parameter PrinterAttributes for the requested printer. Since the IPM Manager is supposed to provide some sort of intelligence, it also retrieves the parameter OtherPrinterAttributes, which are attributes for another appropriate printer. It has been assumed in this description that there are only two printers for black-and-white printing and two printers for colour printing. Moreover, in a real application scenario, there could be more varieties and more complicated types of printers and print server roles. Then a decision should be made. IPM Manager compares the PrinterAttributes with OtherPrinterAttributes, to find out whether PrinterAttributes is better than OtherPrinterAttributes according to the print application's attributes. This decision is done to ensure that the print client has the option to use a more appropriate printer for the print job requested. The actions further taken will depend on the result of the decision. If IPM Manager finds out that the requested printer is more suited to perform the print job, then it sends the Print Application with the signal SendPrintJob to the requested printer. In addition to this it also gives the Print Application its unique Print Job Identification (PrintJobID), which it stores in the Print List described in section 5.4. The IPM Manager then sends the print job identification along with the estimated printing time and printer name to the actor playing the role CSEMon, which stores this information. CSEMon will now monitor the print application in its queue and starts a timer, which equals the estimated printing time it just received from the IPM Manager. After this the actor is said to enter the state Idle. If the decision comes out false, then the IPM Manager sends another proposal to the Print Client with the signal OtherProposal and enters the state WaitForAnswer.

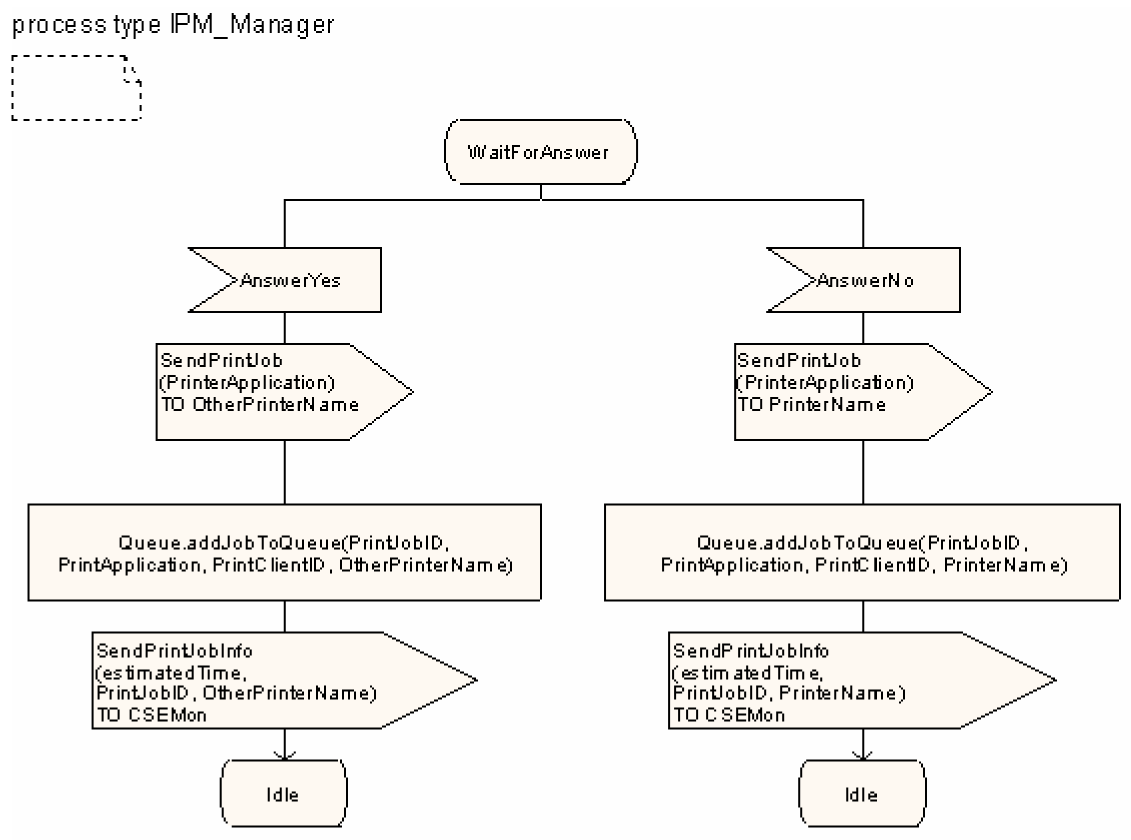

process type IPM_Manager
WaitForAnswer
AnswerYes
AnswerNo
SendPrintJob (PrinterApplication) TO OtherPrinterName
SendPrintJob (PrinterApplication) TO PrinterName
Queue.addJobToQueue(PrintJobID, PrintApplication, PrintClientID, OtherPrinterName)
Queue.addJobToQueue(PrintJobID, PrintApplication, PrintClientID, PrinterName)
SendPrintJobInfo (estimatedTime, PrintJobID, OtherPrinterName) TO CSEMon
SendPrintJobInfo (estimatedTime, PrintJobID, PrinterName) TO CSEMon
Idle
Idle

Figure 21: Part of the manuscript of the IPM Manager

The figure shows the continuation from Figure 20 for the state WaitForAck.

Figure 21 shows the rest of the description when IPM Manager enters the state WaitForAnswer. This state handles two types of messages: AnswerYes and AnswerNo. If AnswerNo is received then it will enter the state Idle. If the print client answers yes, then the same actions as previous are taken. PrinterName is now exchanged with OtherPrinterName.

Each time a print job has emerged, the corresponding print server role sends a confirmation message with the print job identification number to the IPM Manager. The actor playing IPM Manager needs to be able to receive this signal at any state. This is shown for the state Idle in Figure 22.

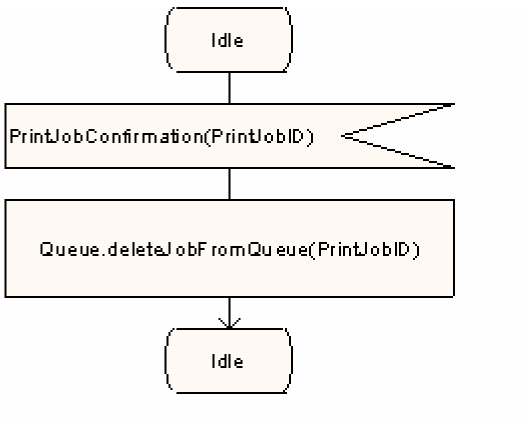

Idle
PrintJobConfirmation(PrintJobID)
Queue.deleteJobFromQueue(PrintJobID)
Idle

Figure 22: Handling print job confirmation messages

IPM Manager receives the signal PrinJobConfirmation with the PrintJobID as parameter. It then it deletes the Print Application from its Print List. IPM Manager re-enters the Idle state.

After studying this example the reader should have good idea on how the manuscripts may be specified using the proposed SDL modelling language.

6 Discussion

This section will discuss the results represented so far. First of all, working with IPM has been challenging due to the complexity of TAPAS. Finding documentation on printing topics, found out to be more difficult than expected. Document research can often result in spending too much time until relevant information is found or not. Full implementation of IPM has therefore not been achieved, taken into consideration of the amount of work with this project. The guidance throughout this project also turned out to be decisive for the results of this project, since important background information was given late and therefore was there not enough time left to fulfil all the goals. Working alone with this kind of projects demands much more effort than working in a team, since it is often easier to progress by motivating each other.

The data models represented have not been tested and may need to be further described in order to let machines comprehend their meaning. The SDL description of the IPM Manager has not been tested, since its purpose is only to show how the different print job processes are done. The SDL diagrams must be translated into XML descriptions, so that machines can understand them.

The Print server role Power Point Master was proposed as an attempt to reduce the number of unnecessary pages that are printed out in large institutions. Other roles with stricter policies and rules regarding document sizes, working hours and types of printer may be defined in order to increase the efficiency in terms of printer usage and achieved tasks.

7 Conclusion

The goals for this project were to specify, design, implement and evaluate IPM using the TAPAS architecture. Different aspects of TAPAS have been studied in order to understand how TAPAS works and how the functionality of IPM could be properly specified according to the TAPAS architecture. An architectural overview over IPM has been proposed along with a complete system description. The required functionality needed to manage the different dynamic configuration aspects has been explained and is consistent with TAPAS Dynamic Configuration Architecture. Some of the data models represented have been extended with new functionality from the existing models, and new ones have been proposed for the handling of predefined printing tasks. The data models were used to describe capability and status descriptions, role specifications and descriptions of configuration and reconfiguration rules. The proposed IPM system handles print jobs in an efficient way. Dynamic solutions are also comprised if a printing problem should occur, and by this preventing print jobs from being lost. Two new print server roles have been proposed, Power Point Master and Picture Master. The latter one is appropriate and easy to use when printing digital photos and graphics, while the former one will help reducing the number of superfluous pages that are often printed.

Further work may be to carry out a full implementation of IPM, which includes the user interfaces, the IPM manager, rules, database, etc. Specification of the informational models should be completed before commencing with the implementation. This implies to handle the representation of both the informational units of the system, e.g. print jobs, print queues, print server, printer capabilities, and the control logic as well as the management rules. The specification and representation of the informational model should be based on the XML notation, making them both human-readable and machine-comprehensible. After this is testing of the implementation required, before demonstrating a network printing scenario that reflect the major characteristics of the solution. Studying the handling of different printer interfaces with different range of capabilities and features might also be an interesting observation.

References

[1] Citation Software Inc. – Specification found at http://ww.citationsoftware.com [Accessed September 2004].

[2] The Columbia Encyclopedia, 6th ed. New York: Columbia University Press, 2002. http://www.encyclopedia.com [Accessed September 2004].

[3] Adobe Systems Incorporated*, PostScript Language Reference Manual*. Addison-Wesley, December 1990. ISBN 0-201-37922-8.

[4] E. Taft, J. Pravetz, S. Zilles, L. Masinter*, The application/pdf Media Type*, RFC3778, Adobe Systems, May 2004.

[5] Adobe Systems Incorporated, PDF Reference, fourth edition. *Adobe Portable Document Format version 1.5.*

[6] Novell*, Introduction to NetWare Print Services*. Specification found at http://www.novell.com [Accessed November 2004].

[7] F-A. Aagensen, C. Anatariya, M. Shiaa, B.E. Helvik*, Dynamic Configuration of Plug-and-Play Systems*. Department of Telematics, NTNU 2003.

[8] T. Hastings, Ed., R. Herriot, R. deBry, S. Isaacson, P. Powell*, Internet Printing Protocol/1.1: Model and Semantics*, RFC2911, Astart Technologies, September 2000.

[9] R. Herriot, I. McDonald*, Internet Printing Protocol/1.1: IPP URL Scheme,* RFC3510, High North Inc., April 2003.

[10] T. Berners-Lee, M. Fischetti and T.M. Dertouzos, *Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web by its Inventor*, Harpus, CA, 1999. http://www.dmtf.org/standards/documents/CIM/DSP0111.pdf [Accessed November 2004].

[11] A. Westerinen and J. Strassner, *Common Information Model (CIM) Core Model*, Version 2.4, Distributed Management Task Force, August 2000.

[12] M. Fowler, K. Scott, *UML Distilled*, Second Edition, Addison-Weasley, 2000. ISBN 0-201-65783.

[13] A. Westerinen and J. Strassner, *Specification for the Representation of CIM in XML*, Version 2.1, Distributed Management Task Force, 2002. http://www.dmtf.org/standards/documents/WBEM/DSP201.html [Accessed November 2004].

[14] C. Anutariya, *XML Declarative Description,* Doctoral Dissertation, Computer Science and Information Management Program, Asian Institute of Technology, Thailand, 2001.

[15] F, Manola, E. Miller, WC3, *RDF Primer*, WC3 Working Draft, 2003. http://www.w3.org/TR/2003/WD-rdf-primer-20030905/ [Accessed November 2004].

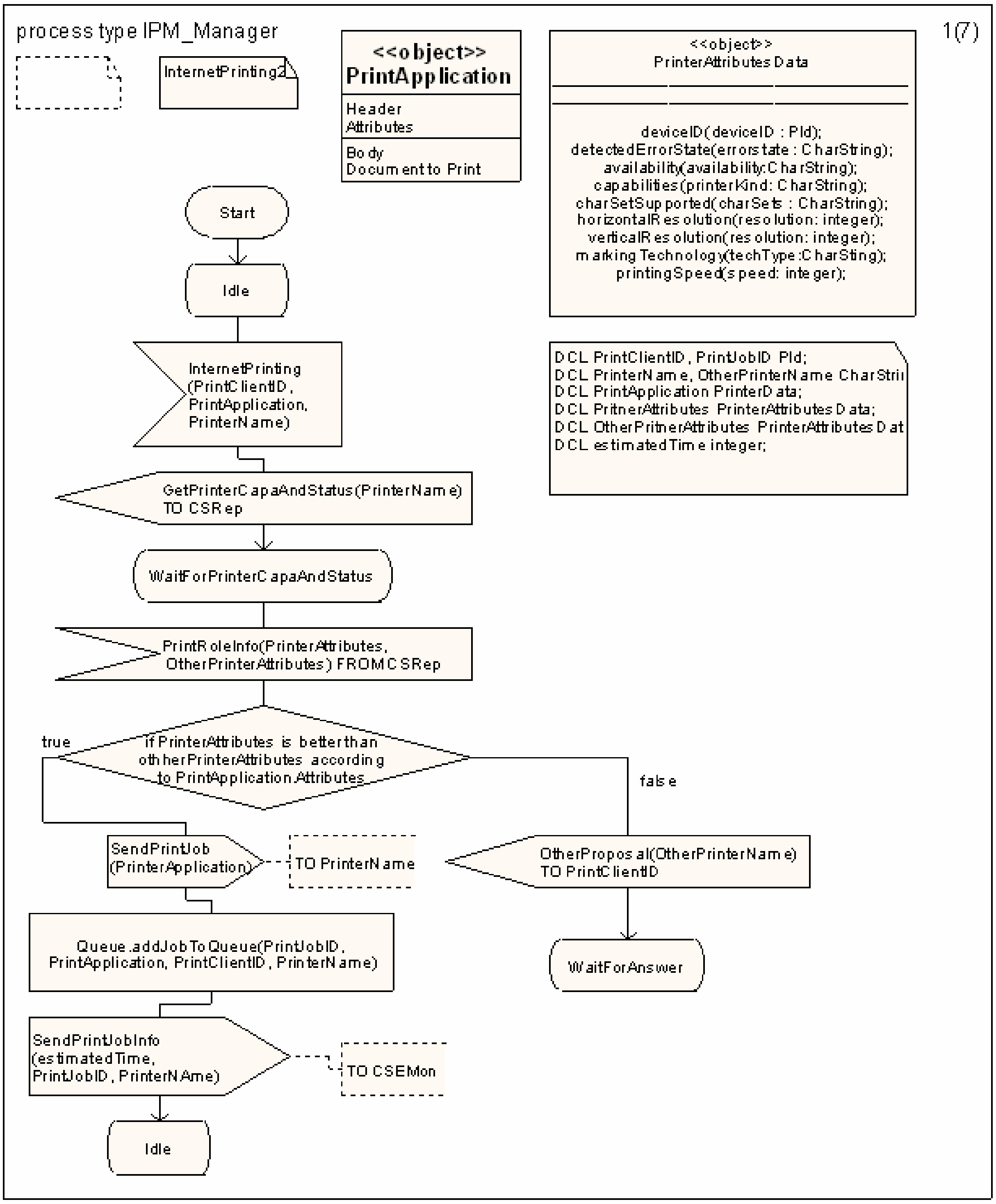
[16] TAPAS. Specification found at http://tapas.item.ntnu.no [Accessed November 2004].

[17] F.A. Aagesen, , B.E. Helvik, V. Wuwongse, H. Meling, R. Bræk, U. Johansen, *Towards a Plug and Play Architecture for Telecommunications*, IFIP Fifth International Conference on Intelligence in Networks (SmartNet99), Bankok – Thailand, 1999. http://www.item.ntnu.no/~meling/publications/papers/smartnet1999.pdf [Accessed November 2004]

[18] R. Bræk, Ø. Haugen, *Engineering Real Time Systems,* Prentice Hall, 1993. ISBN: 0-13-034448-6

[19] F-A. Aagensen, C. Anatariya, M. Shiaa, B.E. Helvik*, On Adaptable Networking.* The First International Conference on Information and Communication Technologies, ICT'2003, Assumption University Thailand, April 2003 http://tapas.item.ntnu.no/publications/ICT2003.pdf [Accessed November 2004]

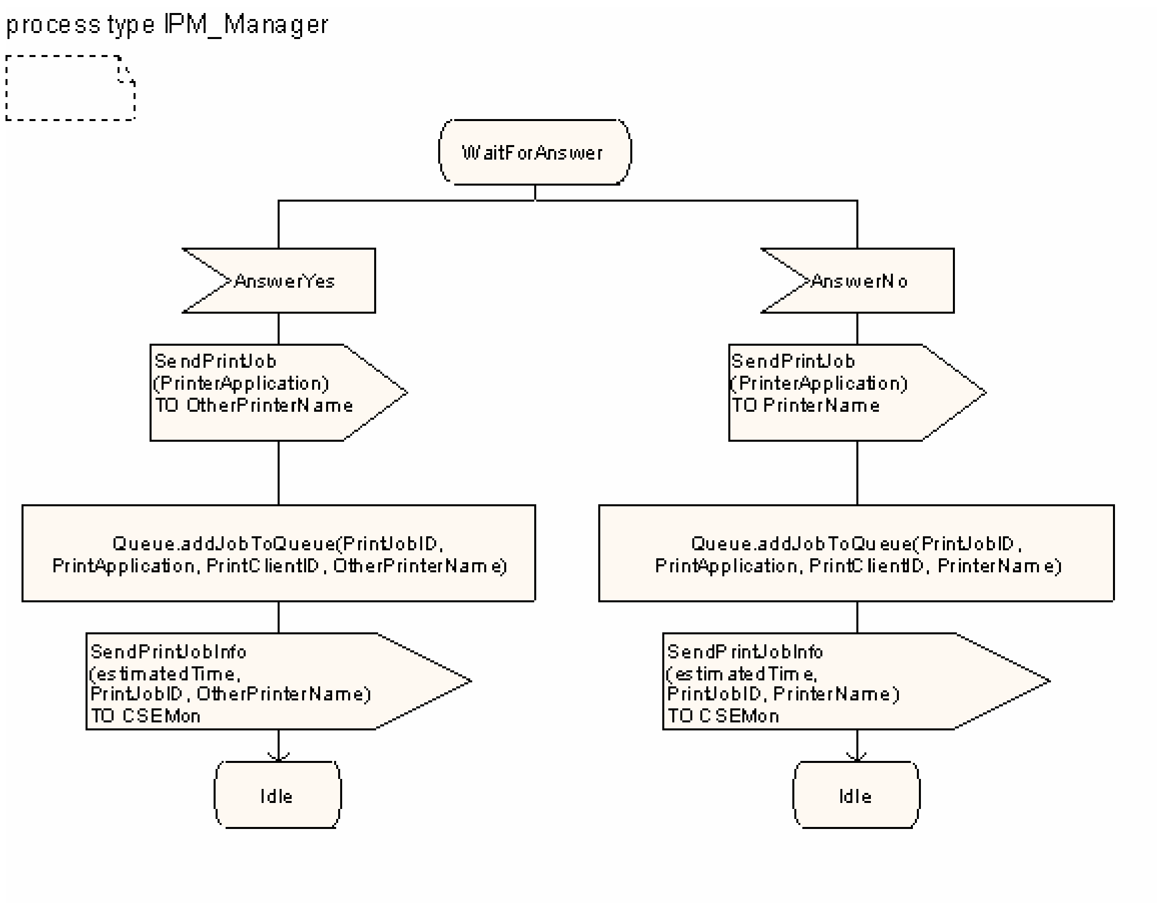
APPENDIX A: Manuscript of the role IPM Manager


process type IPM_Manager
1(7)
InternetPrinting2
<<object>>
PrintApplication
Header
Attributes
Body
Document to Print
<<object>>
PrinterAttributes Data
Start
Idle
InternetPrinting (PrintClientID, PrintApplication, PrinterName)
GetPrinterCapaAndStatus(PrinterName) TO CSRep
WaitForPrinterCapaAndStatus
PrintRoleInfo(PrinterAttributes, OtherPrinterAttributes) FROM CSRep
true
if PrinterAttributes is better than oth herPrinterAttributes according to PrintApplication.Attributes
false
SendPrintJob (PrinterApplication)
TO PrinterName
OtherProposal(OtherPrinterName) TO PrintClientID
WaitForAnswer
Queue.addJobToQueue(PrintJobID, PrintApplication, PrintClientID, PrinterName)
SendPrintJobInfo (estimatedTime, PrintJobID, PrinterNAme)
TO CSEMon
Idle

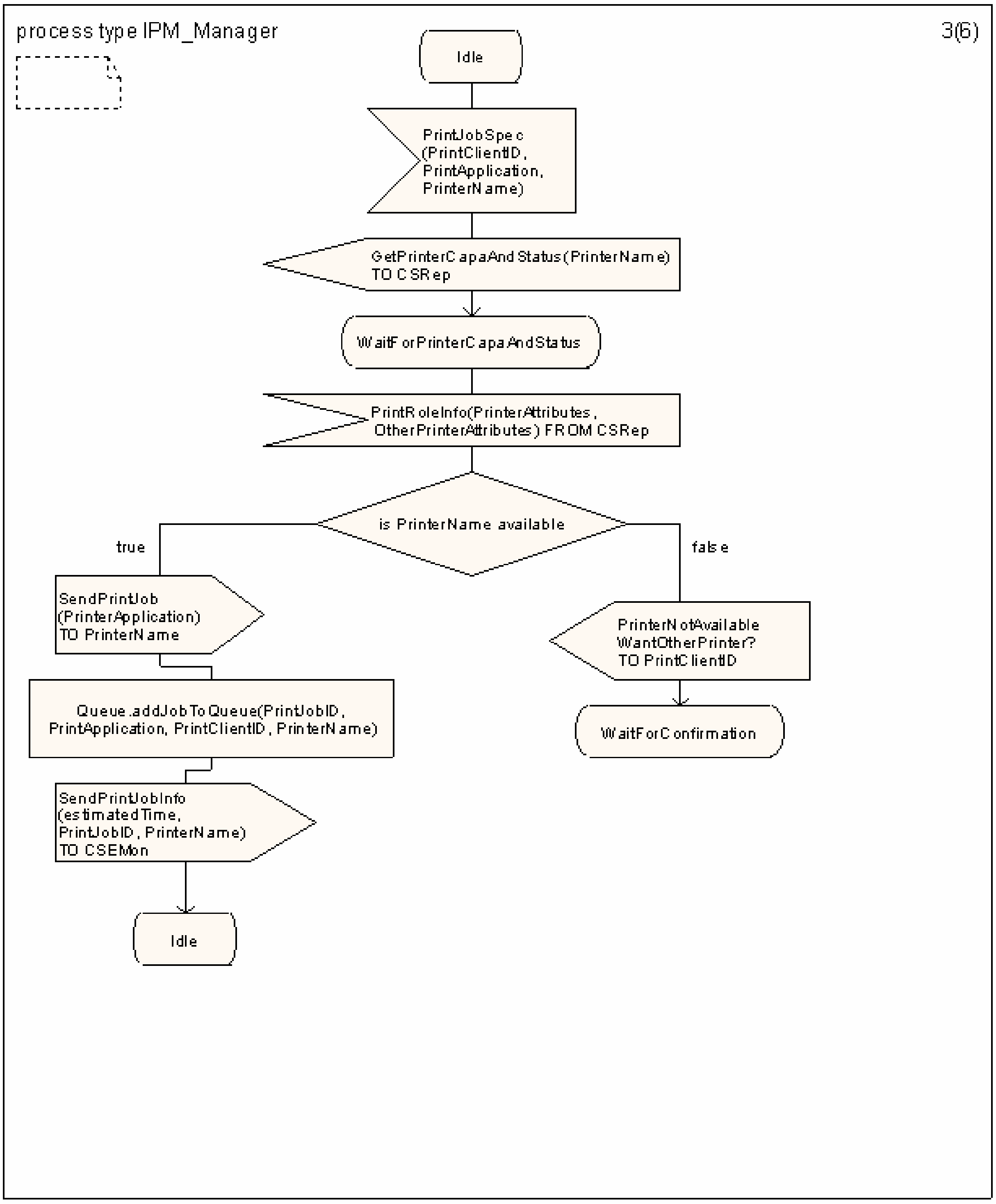
```
deviceID(deviceID : PId);
detectedErrorState(errorstate : CharString);
availability(availability:CharString);
capabilities(printerKind: CharString);
charSetSupported(charSets : CharString);
horizontalResolution(resolution: integer);
verticalResolution(resolution: integer);
markingTechnology(techType:CharSting);
printingSpeed(speed: integer);
```

```
DCL PrintClientID, PrintJobID PId;
DCL PrinterName, OtherPrinterName CharStri
DCL PrintApplication PrinterData;
DCL PritnerAttributes PrinterAttributesData;
DCL OtherPritnerAttributes PrinterAttributesDat
DCL estimatedTime integer;
```

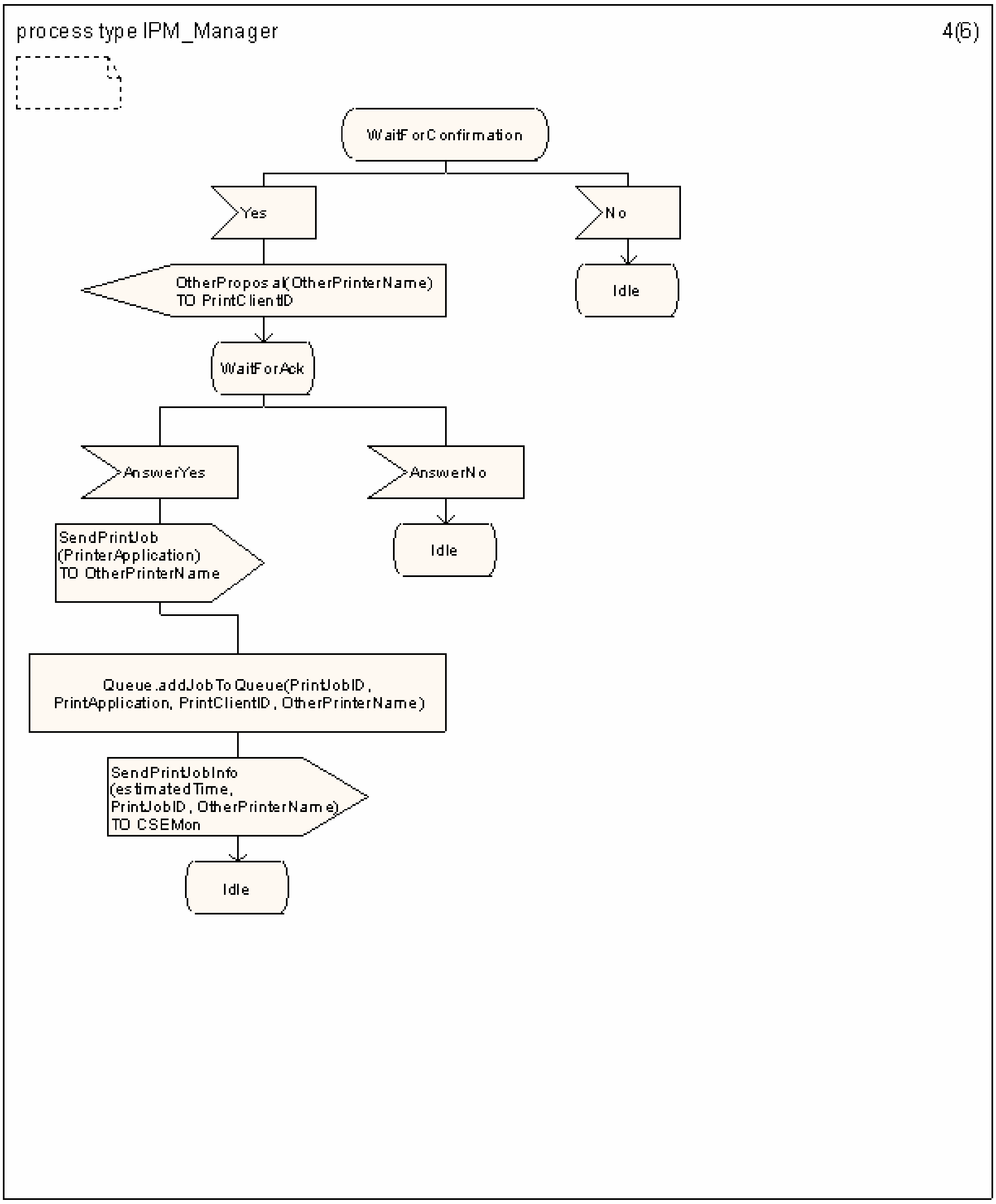
IPM Manager manuscript. Reveiced signal InternetPrinting Part 1



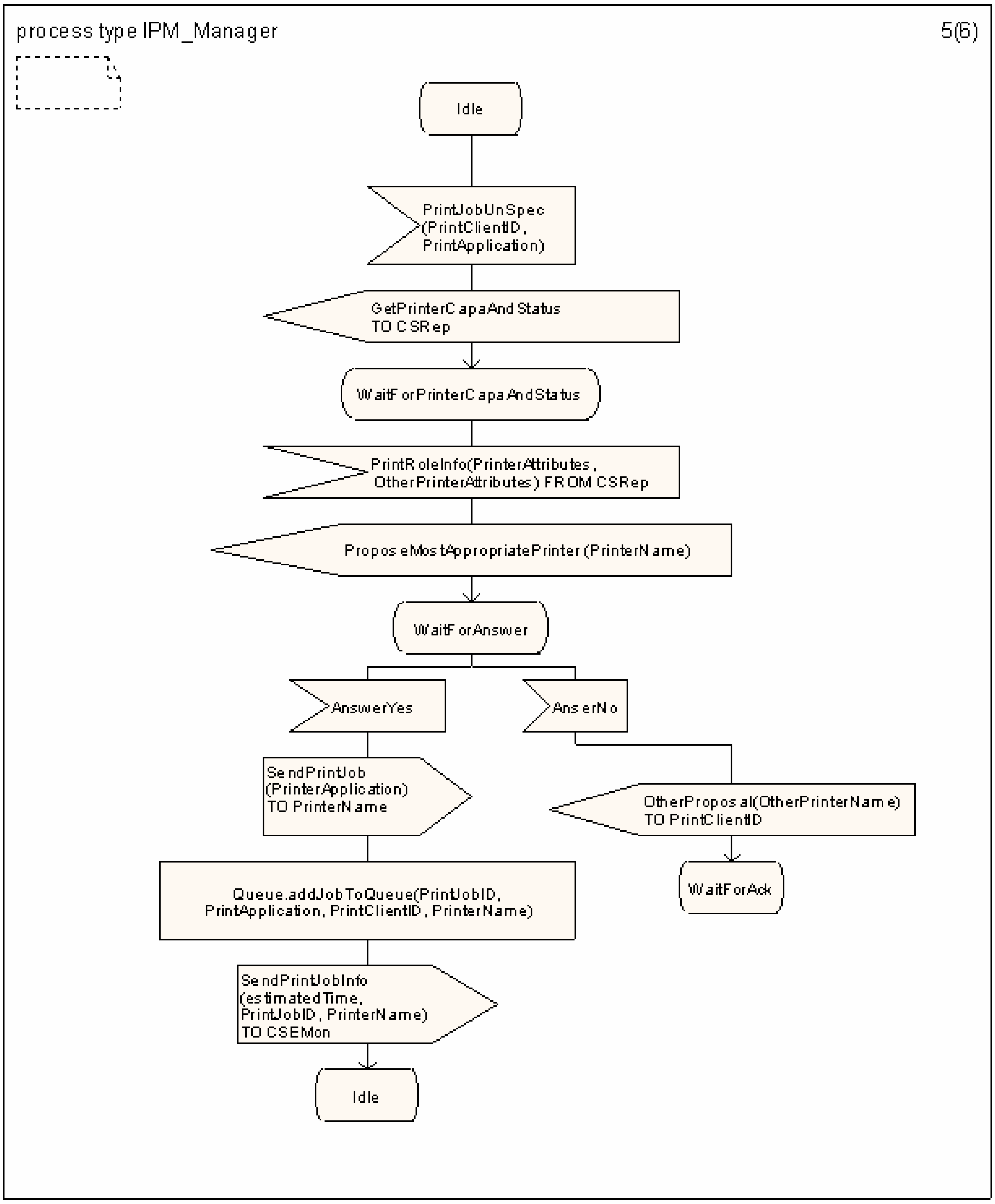
IPM Manager manuscript. Reveiced signal InternetPrinting Part 2


process type IPM_Manager
3(6)
Idle
PrintJobSpec (PrintClientID, PrintApplication, PrinterName)
GetPrinterCapaAndStatus(PrinterName) TO CSRep
WaitForPrinterCapaAndStatus
PrintRoleInfo(PrinterAttributes, OtherPrinterAttributes) FROM CSRep
is PrinterName available
true
false
SendPrintJob (PrinterApplication) TO PrinterName
PrinterNotAvailable WantOtherPrinter? TO PrintClientID
Queue.addJobToQueue(PrintJobID, PrintApplication, PrintClientID, PrinterName)
WaitForConfirmation
SendPrintJobInfo (estimatedTime, PrintJobID, PrinterName) TO CSEMon
Idle

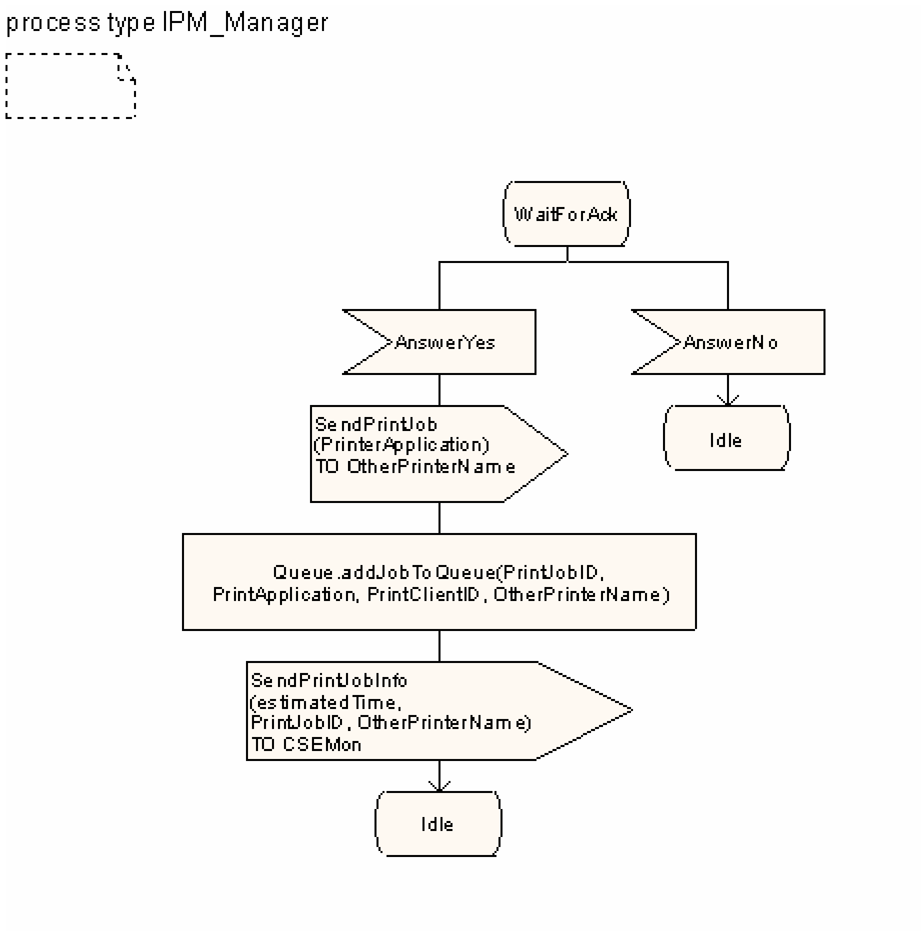
IPM Manager manuscript. Reveiced signal PrintJobSpec Part 1


process type IPM_Manager
4(6)
WaitForConfirmation
Yes
No
OtherProposal(OtherPrinterName) TO PrintClientID
Idle
WaitForAck
AnswerYes
AnswerNo
SendPrintJob (PrinterApplication) TO OtherPrinterName
Idle
Queue.addJobToQueue(PrintJobID, PrintApplication, PrintClientID, OtherPrinterName)
SendPrintJobInfo (estimatedTime, PrintJobID, OtherPrinterName) TO CSEMon
Idle

IPM Manager manuscript. Reveiced signal PrintJobSpec Part 2


process type IPM_Manager
5(6)
Idle
PrintJobUnSpec (PrintClientID, PrintApplication)
GetPrinterCapaAndStatus TO CSRep
WaitForPrinterCapaAndStatus
PrintRoleInfo(PrinterAttributes, OtherPrinterAttributes) FROM CSRep
ProposeMostAppropriatePrinter (PrinterName)
WaitForAnswer
AnswerYes
AnserNo
SendPrintJob (PrinterApplication) TO PrinterName
OtherProposal(OtherPrinterName) TO PrintClientID
WaitForAck
Queue.addJobToQueue(PrintJobID, PrintApplication, PrintClientID, PrinterName)
SendPrintJobInfo (estimatedTime, PrintJobID, PrinterName) TO CSEMon
Idle

IPM Manager manuscript. Reveiced signal PrintJobUnspec Part 1


process type IPM_Manager
WaitForAck
AnswerYes
AnswerNo
SendPrintJob (PrinterApplication) TO OtherPrinterName
Idle
Queue.addJobToQueue(PrintJobID, PrintApplication, PrintClientID, OtherPrinterName)
SendPrintJobInfo (estimatedTime, PrintJobID, OtherPrinterName) TO CSEMon
Idle

IPM Manager manuscript. Reveiced signal PrintJobUnspec Part 2

APPENDIX B: XML role descriptions

XML description of the XML clause PPMaster:

```
<!XML clause PPMaster : capability and status requirements for the role PPMaster>

  <xet:Rule name = "PPMaster" priority = "3">
     <xet:Head>
        <Actor>
           <rolePlaying rdf: resource = "http://PaP.org/PPMaster"/>
           <nodeInstalling rdf:resource = "Svar_deviceID"/>
        </Actor>
     </xet:Head>
     <xet:Body>
        <xer:FactQuery>
        <INSTANCE ClassName = "CIM_Printer">
           <PROPERTY NAME = "DeviceId">
                 <VALUE>Svar_deviceID</VALUE>
              </PROPERTY>
           <PROPERTY NAME = "DetectedErrorState">
                <VALUE>Svar_errorState</VALUE>
              </PROPERTY>
              <PROPERTY NAME = "Availability">
                <VALUE>Running/Full Power</VALUE>
              </PROPERTY>
           <PROPERTY.ARRAY NAME = "Capabilities">
                <VALUE.ARRAY>
                 <VALUE>Duplex Printing</VALUE>
                    <VALUE>Black-and-White Printing</VALUE>
                    <VALUE>Printing 4 and 4 Foils</VALUE>
                 </VALUE.ARRAY>
              </PROPERTY.ARRAY>
              <PROPERTY NAME = "HorizontalResolution">
                 <VALUE>Svar_horizontal</VALUE>
              </PROPERTY>
              <PROPERTY NAME = "VertivalResolution">
                 <VALUE>Svar_vertical</VALUE>
              </PROPERTY>
              <PROPERTY NAME = "MarkingTechnology">
                 <VALUE>Laser</VALUE>
              </PROPERTY>
              <PROPERTY NAME = "PrintingSpeed">
                 <VALUE>Svar_speed</VALUE>
              </PROPERTY>
              <PROPERTY NAME = "CharSetSupported">
                 <VALUE.ARRAY>
                 <VALUE>utf-8</VALUE>
                 <VALUE>us-ascii-8</VALUE>
                 <VALUE>iso-8859</VALUE>
                 </VALUE.ARRAY>
              </PROPERTY>
              Evar_printerProperties
        </INSTANCE>
        </xet:FactQuery>
        <xet:GtE number1="Svar_horizontal" number2="1000"/>
        <xet:GtE number1="Svar_vertical" number2="1000"/>
       <xet:GtE number1="Svar_speed" number2="25"/>
        <xet:NotMember element="Svar_errorState" list="NoPaper NoToner DoorOpen
        Jammed ServiceRequested"/>
     </Body>
  </xet:Rule>
```

XML description of the XML clause PicMaster:

```
<!XML clause PicMaster : capability and status requirements for the role PicMaster>

    <xet:Rule name = "PicMaster" priority = "3">
        <xet:Head>
            <Actor>
                <rolePlaying rdf: resource = "http://PaP.org/PicMaster"/>
                <nodeInstalling rdf:resource = "Svar_deviceID"/>
            </Actor>
        </xet:Head>
        <xet:Body>
            <xer:FactQuery>
            <INSTANCE ClassName = "CIM_Printer">
                <PROPERTY NAME = "DeviceId">
                        <VALUE>Svar_deviceID</VALUE>
                    </PROPERTY>
                <PROPERTY NAME = "DetectedErrorState">
                      <VALUE>Svar_errorState</VALUE>
                    </PROPERTY>
                    <PROPERTY NAME = "Availability">
                      <VALUE>Running/Full Power</VALUE>
                    </PROPERTY>
                <PROPERTY.ARRAY NAME = "Capabilities">
                      <VALUE.ARRAY>
                            <VALUE>Simplex Printing</VALUE>
                            <VALUE>Colour Printing</VALUE>
                        </VALUE.ARRAY>
                    </PROPERTY.ARRAY>
                    <PROPERTY NAME = "HorizontalResolution">
                        <VALUE>Svar_horizontal</VALUE>
                    </PROPERTY>
                    <PROPERTY NAME = "VertivalResolution">
                        <VALUE>Svar_vertical</VALUE>
                    </PROPERTY>
                    <PROPERTY NAME = "MarkingTechnology">
                        <VALUE>Laser</VALUE>
                    </PROPERTY>
                    <PROPERTY NAME = "PrintingSpeed">
                        <VALUE>Svar_speed</VALUE>
                    </PROPERTY>
                    <PROPERTY NAME = " GraphicsFileSupported">
                        <VALUE.ARRAY>
                            <VALUE>GIF</VALUE>
                            <VALUE>JPEG</VALUE>
                            <VALUE>BMP</VALUE>
                            <VALUE>TIFF</VALUE>
                            <VALUE>TGA</VALUE>
                            <VALUE>RAS</VALUE>
                            <VALUE>EPS</VALUE>
                            <VALUE>PCX</VALUE>
                            <VALUE>WMF</VALUE>
                            <VALUE>PNG</VALUE>
                        </VALUE.ARRAY>
                    </PROPERTY>
                    Evar_printerProperties
            </INSTANCE>
            </xet:FactQuery>
            <xet:GtE number1="Svar_horizontal" number2="4000"/>
            <xet:GtE number1="Svar_vertical" number2="1200"/>
            <xet:GtE number1="Svar_speed" number2="15"/>
            <xet:NotMember element="Svar_errorState" list="NoPaper NoToner DoorOpen
            Jammed ServiceRequested"/>
        </Body>
    </xet:Rule>
```