14

Software design practice using two SCADA software packages.

K.P. Basse ,G.K. Christensen and P.K. Frederiksen
Dept. of Control and Engineering Design
Technical University of Denmark
Building 421, DK-2800 Lyngby
Phone: +45 4525 4504, Fax: +45 4288 4024

Abstract

Typical software development for manufacturing control is done either by specialists with considerable real-time programming experience or done by the adaptation of standard software packages for manufacturing control. This paper covers the experiences from the application of two software packages for Supervisory Control And Data Acquisition for the control of a chemical plant. The software packages are: "Fix" from Intellution and "InTouch" from Wonderware. Emphasis is placed on the practical software design cycle and the typical changes imposed on this by the application of the standard packages. It is argued that the SCADA and the PLC software must be developed in relation to a common specification for both applications although the development is done in two very different environments. Comparison of the software packages in relation to the process control specifications and the development effort is given.

Finally the application possibilities for these packages in relation to discrete parts manufacturing are evaluated based on the internal structure of the packages and the structural demands in this area.

Keywords

Supervisory control, standard SCADA software, software design.

1 INTRODUCTION

Ever since the introduction of the CIM-concept there has been a general consensus that the computer integrated factory is the factory of the future (with a future !) . No matter what specialised mode of operation we are considering there has been increased efforts to use computers both for off-line planning and on-line production control. Inside the discrete parts manufacturing area the scene has been dominated by special purpose controllers like CNC-controllers, robot-controllers , more general process controllers PLC-controllers, etc. The process industries have relied more heavily on general process controllers and PLC's. Both areas face the challenge to integrate off-line planning with on-line production control. In

A. Storr et al. (eds.), *Software Engineering for Manufacturing Systems*

general this challenge is expressed in terms like optimised production scheduling, quality control, product documentation and responsibility, CAD/CAM integration, etc. In both areas considerable integration has been demonstrated in terms of the functionality of the implemented plants. The demonstrations have mainly been achieved by completely new plants where one or two vendors have supplied everything from computer hardware and software to production machines or PLC's. The computer integration of manufacturing in companies with existing production facilities seems to go much slower due to the large investment in software development involved. One of the reasons for this is the lack of industrially accepted standards for the areas and the large amount of work involved in achieving commonly accepted goals. These goals may be summarised as (Fix, 1994; InTouch, 1995; Mainstream, 1989):

* independence of hardware platform
* high level programming
* drivers to a large diversity of special hardware
* easy to set up graphical man/machine interface
* database access
* network capabilities
* transaction processing meeting real-time demands
* manufacturing specific support functions

On the software market there seem to be an increasing number of standard software packages available for manufacturing control. Some of these are termed SCADA software. SCADA is an abbreviation of : **S**upervisory **C**ontrol **A**nd **D**ata **A**cquisition. They are aimed primarily at various forms of process control and data acquisition tasks. Two successful packages that seem to fulfil the above mentioned general requirements were selected for the control of a typical chemical process plant. These were: Fix (Fix,1994) and InTouch (InTouch, 1995).The purpose was to test the packages in PC-versions against a specific set of process control specifications and to study the software design methods and related programming effort involved in the use of these packages.

2 THE SYSTEM SPECIFICATIONS

A simplified PI-diagram of the chemical chromatographic plant is shown in figure 1. Chromatography is a term used for a number of processes where different components of a mixture are separated in a process column by passing various chemical substances from input tanks through the column. The process does not demand real-time responses with reaction times of less than 1 second. In this respect the PC based software packages seemed not to present any problem. Nevertheless the PC is not used to directly control the plant. Because of high demands for control security and related pharmaceutical validation, the process has been controlled by a stand alone PLC .(SattControl,1995). The PLC contains the interface hardware for the process instrumentation and control valves. For the above mentioned reasons the structure for the plant control with SCADA software became a single PC/PLC configuration as shown in figure 2.
Additionally there was a demand that the PLC should be able to control the process also in the case where the PC might fail. This limits the SCADA software to the task of supervisory control and data acquisition. A short description of the control system specification is given below.

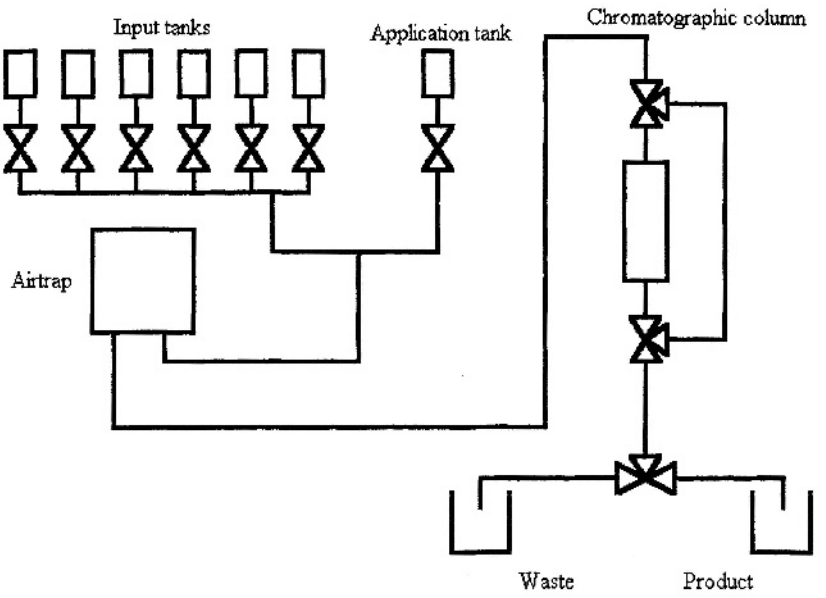

Figure 1 Simplified PI-diagram for the chromatografic test plant.

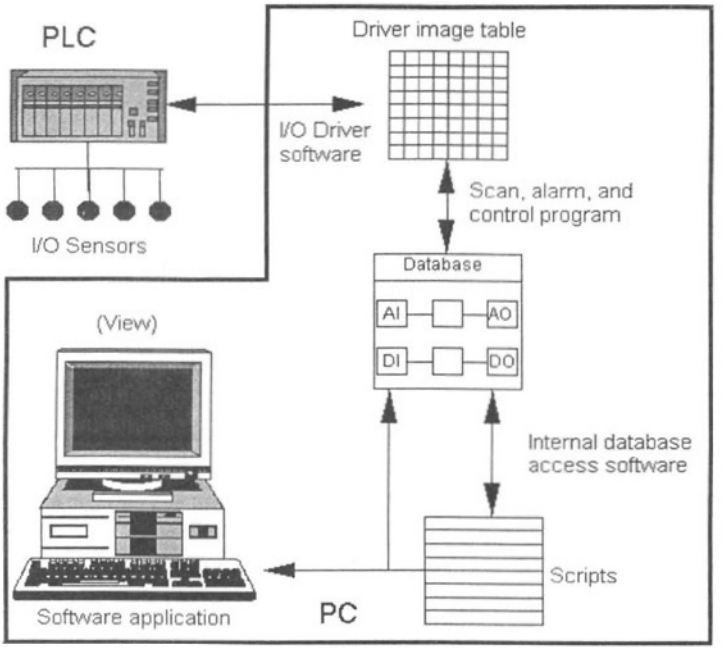

Figure 2 PLC and PC interface structure.

The system will be in one of four states: Manual, Running, Idle or Error.

Manual: A state where the sequences of operations are all handled manually.

Running: A normal operating state where the PC/PLC -control system is running a number of batches. For each batch an initialisation is carried out, where the complete batch description is downloaded from the PC to the PLC.
Operator acknowledgements of alarms.
General data acquisition, logging and plotting should be done for two purposes, -one for on-line information on the PC-screen and one for historical documentation and off-line analysis of data stored in a database.
It should be possible to override the downloaded sequences by manual intervention to a certain degree, - for example to suspend normal running by entering an idle state.

Idle: A number of security actions are taken and further sequencing is halted.

Error: A state which only the system itself can generate according to a number of specific error conditions.

As can be seen from the foregoing description the system specification consists of a mixture of plant specifications and plant control specifications. The plant control related specifications are initially divided into a PLC-control specification and a SCADA- supervisory control specification for the control software design. In carrying out this division a specific effort is taken to clarify the interacting parts of the specification that is related to the communication between the controlling PC and PLC. This part of the specification is a result of the decision to divide the control task in a PLC and a PC-task. An important aspect is that any function required at the PC level that needs to access the process parameters can only get information or execute control insofar as these functions are supported by the PLC control software. Although the development was done by different programmers on the PC and the PLC level, it turned out that a close cooperation between these was necessary to allow a smooth transition into an integrated system. The advantage of using different programmers for the two control software tasks is that a higher degree of specialisation can be achieved. This is desirable taking into account the very different environments of programming and corresponding strategies that exist at the two levels. These methods and strategies will be dealt with in the following.

3 FUNCTIONALITY OF THE SCADA SOFTWARE PACKAGES

As the abbreviation SCADA indicates, the main objective of the software packages is Supervisory Control And Data Acquisition.
* *Data acquisition* is the ability to retrieve data from the plant floor and process this data into a useful form. Data can also be written to the plant floor, thereby establishing the critical two-way link that closed-loop control software require. The data transmission details of the communication with the process hardware are handled by an I/O-driver. The I/O-driver is selected for the hardware used in the plant, e.g. for the specific PLC used.
* *Supervisory control* is the ability to monitor real-time data coupled with the ability of operators to change set points and other key values directly from the computer. This function is typically achieved via a graphic interface which is constructed with the SCADA software packages in relation to the structure of the plant under control.
The tasks above are usually handled by the SCADA software package's central elements which are the graphical user interface, the database, and the I/O driver. A more detailed description of these elements is given in the following chapter, where the usage is illustrated by a simple example using the InTouch and Fix packages.

3.1 Other functions often found in Scada software packages

* *Data management* is the process where the acquired data is manipulated according to the requests of the software applications constructed by the user. The basic functions of data acquisition and management provide the basis for practically all the industrial automation tasks that the SCADA software packages can perform. The absolute *data integrity* is therefore an essential demand when automation is done with the help of these packages. Most of the products have some kind of tool to assist to ensure this integrity.
* *Monitoring* is the ability to display real-time plant floor data to operators. Powerful numeric, text, and graphical formats are usually available to make data more accessible.
* *Alarming* is the ability to recognise exceptional events and immediately report those events to the operators via the graphical display.
* *Control* is the ability to automatically apply algorithms that adjust process values and thereby maintain those values within predefined limits. Control goes one step beyond supervisory control by removing the need for human interaction. The computer can thereby be used to control the whole process or a part of the process to be automated.
* *Data archiving* is the ability to sample and store any data point in the system in data files at operator specified rates. At any time the data can likewise be retrieved from the data files to create trend displays of historical data. Managers and engineers can use this data to examine the events leading up to a critical event after addressing more immediate problems. The archived data represents a powerful tool for process correction and optimisation.
* *Reporting* is the possibility to access process data through industry standard data exchange protocols such as DDE(Dynamic Data Exchange) or ODBC SQL (Open DataBase Connectivity Structured Query Language). Operators can thereby create detailed reports with spreadsheets, compatible with MS Excel or MS Access, that contain acquired and calculated historical data.
* *Distributed processing* is the ability to create a system where the processing is distributed over a network.
* *Centralised processing* is the ability to have a system that contains only one computer (node) i.e. stand alone system
* *Time-based processing*

Most applications work by acquiring and calculating data at regular intervals, defined in seconds, minutes, or hours. Most SCADA software packages can perform any combination of *time-based processing*. This function allows the operator to balance the system resources giving priority to data that needs to be acquired quickly.

**Exception-based processing*

Processing that is triggered by events rather than time is known as *exception-based processing*. Processing can here be triggered by data changes, unsolicited messages from the process hardware, operator actions or other software applications. *Exception-based processing* is an essential feature for achieving distributed SCADA applications that monitor a large number of I/O devices.

4 TOOLS FOR PROGRAMMING THE SCADA APPLICATION

4.1 Introduction

During the construction of a SCADA application, the implementer encounters three different types of programming. These programming types are found when designing the graphical user interface, when setting up the database of the system, and when programming scripts. In the following sections these programming types are described in relation to Fix and InTouch, with

some illustrative figures generated from a simple example. The example used is a 15000 litre horizontal tank where the volume in relation to the liquid level should be calculated and illustrated graphically. The input/output is volume/level or vice versa. The graphical user interface of this system is shown in figure 3.

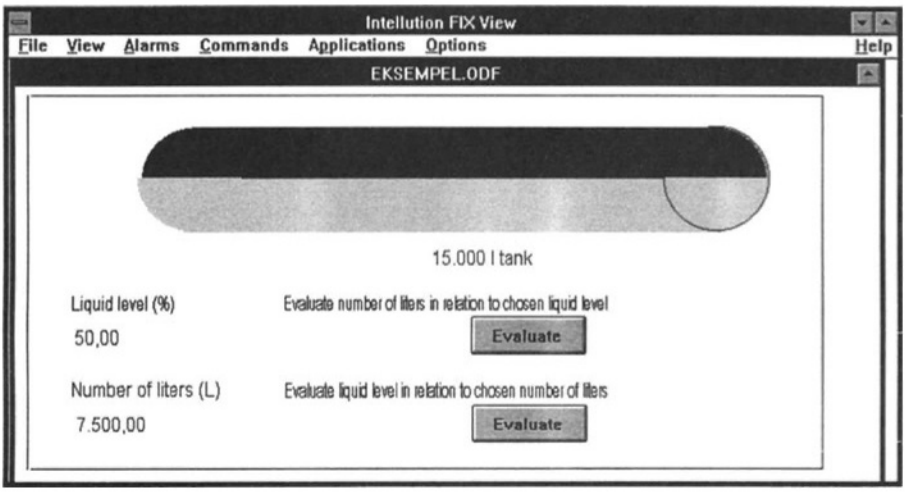

Figure 3 Graphical interface for a horizontal tank. (Fix, 1994)

4.2 Graphical user interface

When constructing the graphical user interface, the SCADA packages provide some helpful tools for designing the graphic display. The most important is a toolbox where many possibilities for drawing and creating the computer displays, that allow the operator to interact with the process data are found. Another useful feature is the creation of smaller elements that are often used. These could be valves, tanks, etc. These elements are called Wizards in InTouch and Dynamoes in Fix. When designing the graphical user interface the constructor can re-use his set of predefined Wizards/Dynamoes and he will be asked to fill in the properties for each element. This is a powerful tool because it is possible to create a large library of standard elements that is very useful if there are strict demands for validation. The toolbox from Fix can be seen in figure 4.

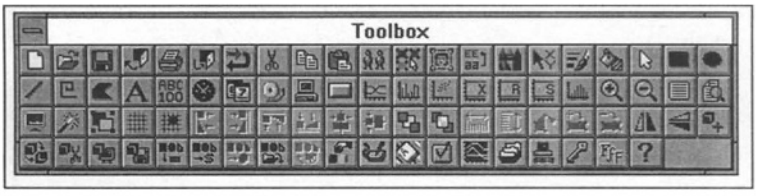

Figure 4 Toolbox for programming via dynamoes (Fix, 1994)

4.3 Database programming

As mentioned above, the database is one of the central parts of the SCADA packages. The database of a SCADA package creates the connection between the process hardware via a

driver to the graphical user interface via internal database access software as shown in figure 2. The first step in creating the database for the system to be controlled is configuring the driver that connects the database to the process hardware of the system that, as mentioned above, is typically a PLC. The driver is usually accessed by two different approaches - poll or event based, where Fix uses a poll-based driver and InTouch uses an event-based driver. A poll based driver continuously updates the data points whereas an event based driver only updates if there has been a significant change of the value in question. The implementer has to create driver image tables or poll tables where each I/O point from the PLC that is to be used in the database must be defined. Here the implementer must specify how often a data point is to be updated, which type of data is used, (whether is analog, digital or a register) and the specific address in the PLC. The drivers used by the two SCADA packages in question were not only different concerning the database updates. The driver used by Fix was a DOS program and the driver used by InTouch was a Windows program. As a result of this fact, Fix had to be restarted each time changes were made in the driver in order to acknowledge the changes, which was quite annoying. Once the driver image tables were completed the next step was to configure the database of the SCADA package. During configuration the main objective was to connect the driver image table to the graphical user interface. The first step was to decide what kind of variable, called a *tag*, was to be used. When this decision was made the implementer would be presented with a small window that is programmed via fill-in-the-blanks as seen for Fix in figure 5. From here it is possible to set up the hardware connection to the PLC, specify ranges and alarm values and so forth. When the database is fully equipped with all the connections to the hardware, the data that it contains is ready to be used in the graphical user interface direct or via scripts as explained in the following section. Values of the database can also be sent from the PC to the PLC. The Fix database builder for the example is shown in figure 6.

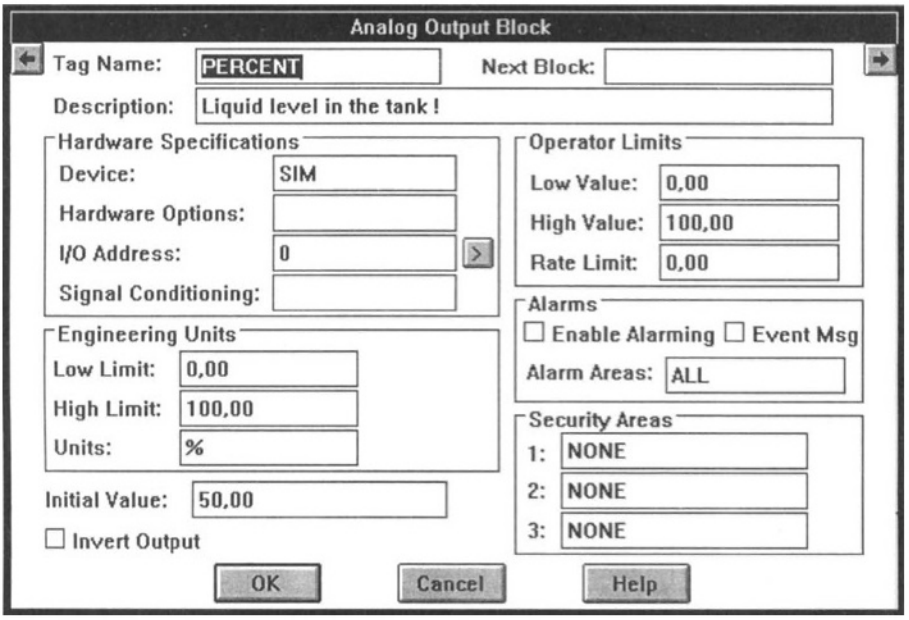

Figure 5 Fill-in-the-blanks programming (Fix).

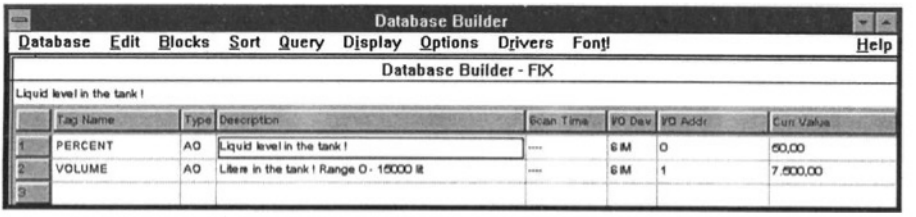

Figure 6 Database builder (Fix).

4.4 Script programming

The final step in constructing the application is to use the database tags in scripts, which are small programs that often initiated by an operator. These programs can alternatively be started by events taking place in the application. The scripts are used for manipulating data from the hardware so it is possible to present these in just the right way in the graphical user interface. The programming tools for programming the scripts are for Fix and InTouch quite different. Fix uses a programming language, called Command Language, where most of the commands are specific for the Fix package. The programming language used by InTouch is more similar to standard languages like Fortran and Pascal. Another important difference is the size of the programs. Fix only has the possibility to make scripts containing 50 program lines, where InTouch support almost an infinite number of program lines. This makes script programming in Fix very difficult because space is very limited. Both of the mentioned SCADA packages have promised changes in this field, and future releases of the script language will be based on Visual Basic. Because the Fix package uses a special set of commands it is time-consuming for a novice to construct perfect scripts. In figure 7a and figure 7b the scripts used in Fix and InTouch to handle the examples above are shown to give an idea of the differences.

```
1 DECLARE #VOLUME NUMERIC SCRIPT
2 DECLARE #TEMPVOLUME NUMERIC SCRIPT
3 DECLARE #HEIGHT NUMERIC SCRIPT
4 DECLARE #PERCENT NUMERIC SCRIPT
5 DECLARE #I NUMERIC SCRIPT
6 GETVAL FIX:VOLUME.F_CV #VOLUME
7 #TEMPVOLUME = 7500
8 #PERCENT = 50
9 IF #I <= 20
10  GOTO 15
11 ENDIF
12 IF #I > 20
13  GOTO 29
14 ENDIF
15 IF #VOLUME == #TEMPVOLUME
16  GOTO 29
17 ENDIF
18 IF #VOLUME > #TEMPVOLUME
19  #PERCENT = #PERCENT + 50 / (2 ^ (#I + 1))
20  #I = #I + 1
21 ENDIF
22 IF #VOLUME < #TEMPVOLUME
23  #PERCENT = #PERCENT - 50 / (2 ^ (#I + 1))
24  #I = #I + 1
25 ENDIF
26 #HEIGHT = #PERCENT * 17.5
27 #TEMPVOLUME = 2 * 6236,2753 / 1000000
  * ((#HEIGHT / 2 - 437,5) * SQRT(-(#HEIGHT ^ 2)
  + 1750 * #HEIGHT) - 382812,5 *
  ASIN(1 - (2 * #HEIGHT) / 1750)
  + 191406,25 * 3,1425926535)
28 GOTO 9
29 SETVAL FIX:PERCENT.F_CV #PERCENT
```

Figure 7a Script example from Fix for the horizontal tank.

```
1  Volume = VolumeWindowValue;     {Volume first altered when "OK"}
2  TemporaryVolume = 7500;          {First guess, tank half full}
3  TemporaryPercent = 50;
4  FOR i = 0 TO 20                  {Max. 20 iterative actions}
5  IF Volume == Round(TemporaryVolume,1) THEN
6     EXIT FOR;    {Exit if calculated volume equals the volumen given}
7  ENDIF;
8  IF Volume > TemporaryVolume THEN
   {If the calculated volume is smaller than the given volume}
9     TemporaryPercent = TemporaryPercent + 50/(2**(i+1));
   {New guess is calculated}
10 ELSE
   {If the calculated volume is bigger than the given volume}
11    TemporaryPercent = TemporaryPercent - 50/(2**(i+1));
   {New guess is calculated}
12 ENDIF;
13 Height = TemporaryPercent * 17.5; {Height of tank is 1750mm}
14 TemporaryVolume =
   {Calculation of volume corresponding to new TemporaryPercent}
   2*6236.2753/1000000*(((Height)/2-437.5)*Sqrt(-(Height)*(Height)+1750*(Height))
   -382812.5*ArcSin(1-2*(Height)/1750)*3.1415926535/180+191406.25*
   3.1415926535);
15 NEXT;
16 Percent = Round(TemporaryPercent,1);
17 Hide "Pop-up window for volume";
```

Figure 7b Script example from Intouch for the horizontal tank.

4.5 Application programming practice

When programming a new application with the assistance of SCADA packages, it was expected that the approach would vary with the amount of programming experience of the programmer. An experienced programmer will often find the top - down/bottom - up principle quite helpful. This method results in an approach where the total specifications for automation of the system are generated according to the top-down principle mentioned above. Afterwards the actual programming can begin. The code writing then follows the bottom-up principle, where programming starts at the PLC. Once the PLC program is completed, the systems I/O driver is configured. This is done by defining all the I/O points to be used in the application. The next step in the process is to configure the database of the SCADA package according to the I/O points defined in the driver, and the needs represented by the graphical user interface. When these tasks are completed, the graphical user interface is designed in relation to the wishes and demands presented in the specifications. The user screens are mostly constructed with the aim to have some kind of resemblance with the physical system to be controlled. The final process is to connect the elements of the graphical user interface with the database directly or via programming scripts. At this stage the dynamics of the system are made accessible to the operator. The approach presented above is an ideal approach which for beginners was found quite hard to follow. During the programming it was experienced that it was often necessary to move back and forth between the levels as the experience grew. This movement was however not felt by the programmers as a difficulty, because the path from implementation to test was assisted greatly by the software packages. A useful approach here was to advance the graphic presentation part of the software in order to present the planned functionality and to use it as a basis for discussion with the users.

5 TOOLS FOR PROGRAMMING THE PLC

A number of methods for creating PLC-code were identified: Instruction List (IL), Ladder-Diagrams (LD), Function Block Diagrams (FBD) and Sequential Function Charts. (SFC), (IEC 1131-3). What we needed was a high level language and graphical presentation that was easy to present and discuss with plant users. We chose the SFC method, sometimes referred to

as GRAFCET (B. Goran, 1991) which is a graphical method that is excellent for the sequential part of the control, but does not describe data management aspects of the PLC program. The PLC selected for the process was a SattControl 05-45 PLC with 65 Kbytes of memory and could be programmed by a PC software package: DOX-5/10. Because an old version of the software has been written using DOX-5, this version was used. The PLC is equipped with 64 digital I/O points, 8 analog inputs and 4 analog outputs. Even though DOX-5 did not support GRAFCET programming, the method was used throughout to discuss specifications with users and as a specification for the programmer. An example of a list of sequential steps using GRAFCET is shown in figure 8.

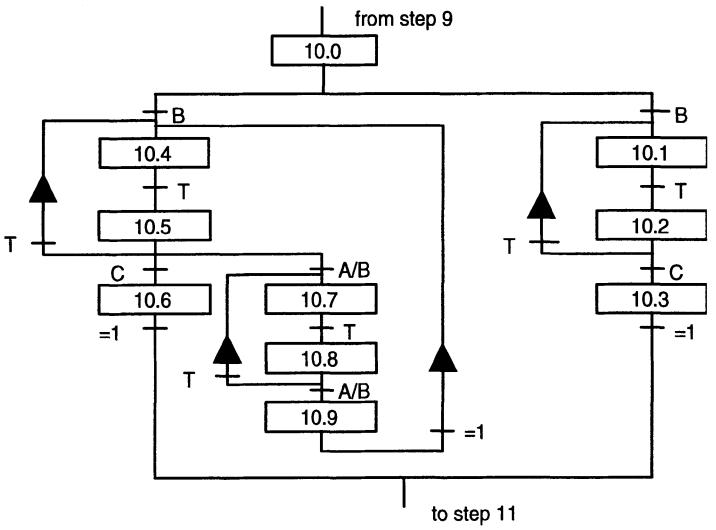

Figure 8 Example of a Sequential Function Chart from the test plant (GRAFCET)

The sequential function chart specifies the central part of the control software, and the other parts are more or less derived from, or merge into, ist states. The output does not normally change without an initial change of state and the states are the basis for forming the output. As additional guidelines for programming, a set of company and PLC specific function blocks and guidelines were used. The function blocks dealt with: self-locking states, sequential steps, component controls and calculations. The guidelines for programming addressed naming conventions, structuring of code and other hints for writing good, serviceable software code. Despite the attempts to keep programming at a standard level this was not totally possible with the PLC-code. Like most PLC applications, the coding was sometimes rather awkward and vendor specific when we got beyond the basic Boolean operations programming. An example is a case where a user demand results in a corresponding demand for flexibility of the PLC program. In the test plant the user wanted to be able to change the set-up of input and output tanks from one GRAFCET state to the next (see figure 1). The solutions suggested were:

a) download a complete PLC-program containing the new input/output specification

b) download a new input/output specification for each change of state

c) download a complete input/output specification for all states

The first alternative conflicts with the desire to keep validation at a minimum, using only one code variant. The second conflicts with the specification in saying that the PLC should be capable of controlling the process in a stand-alone mode. The third method is the only one acceptable. It requires however a set of register and addressing capabilities that would be easy to handle in, for example PASCAL or C, but requires a PLC-code specialist using, e.g. indirect addressing. The result is that a lot of vendor-specific programming tricks are required and the programming for just this vendor product line becomes both time consuming and costly. This again led to a company policy to limit the number of different PLC vendors in order to keep programming and maintenance problems at a manageable level.

6 EXPERIENCES GAINED

In 12 months two beginners of both SCADA and PLC-programming and with only basic PASCAL programming experience set up specifications for and implemented one PLC-program and two SCADA solutions using FIX and InTouch. The specification was a renewed specification for the complete PC/PLC-solution based in part on an existing PLC-specification and program. The new specification took about 6 months in close cooperation with process and plant engineers and operating personnel. The specification process was as usual an interactive process trying to assemble loose ends, harmonise different views and scale ambitions. In the same period, the students became acquainted with the software products at hand: The PLC-programming environment and the two SCADA products. After another 6 months one complete, working process control system was demonstrated and the accompanying documentation finished. The effort is described as 50 % PLC-programming and 50 % SCADA programming.

During implementation some minor problems with the SCADA products were found. The FIX package did not allow for a specified automatic lock-out in the case of inactivity. Concerning the database interface some minor problems were detected when trying to export data from InTouch to the Microsoft database Access. Furthermore it was felt that the SCRIPT language for FIX was less intuitive than the more general purpose SCRIPT language for InTouch. Summarising, the SCADA packages had several ways to facilitate high level programming: Fill-in-the-blanks, wizards-programming and clip arts for the graphics interface. However troublesome these programming tools were felt by the programmers, there is no way they could have made the same kind of control system without this extensive programming support.

Moreover the features of the packages that were not tested or used presently for the process control are still available for future extension and integration in the company. Thus path to "open" systems using de facto standards like Windows, DDE, NetDDE and ODBC SQL achievable. One important aspect of programming was not carried out during the above mentioned period - the final system validation. The validation procedures followed by the company is regulated to a large extent by the pharmaceutical regulations. The validation process was estimated to take an additional 3-6 months. (R. Konakovsky, 1994). The uncertainties relating to the validation of the large software package running under Windows further underlined the necessity for a solution where the critical parts of process control could be run by the PLC alone.

7 CRITICAL ISSUES

The application of standard software speeded up the development of the application, but also restricted the freedom and insight of the programmer. In this case, as in many real-time control applications, it is almost impossible for the programmer to estimate the time required in data acquisition, data management and data presentation. In this case no extreme demands for event management and small scan-times were specified. The fastest scans were measured in seconds and minutes. Concluding, it would be very useful to have a set of standard tests or benchmarks for different configurations of SCADA products and PLC's. In this respect, special attention must be given to the driver and communication hardware.

8 FUTURE ASPECTS

The software development described leads to the conclusion that considerable development time can be saved by integration of the two software tasks for the PLC and the SCADA application. The integration is illustrated in figure 9. Here the PC and the PLC melt into a "SoftPLC". The advantage is that the construction of the PLC-code can automatically build the database for the SCADA application. Furthermore, the interaction between the PLC and SCADA functions are optimised, not dependent on different drivers. Another feature illustrated by figure 9 is the possible change of the process instrumentation interface, which can be shifted towards a fieldbus solution, where all sensor signals are sent via the network in digital form.

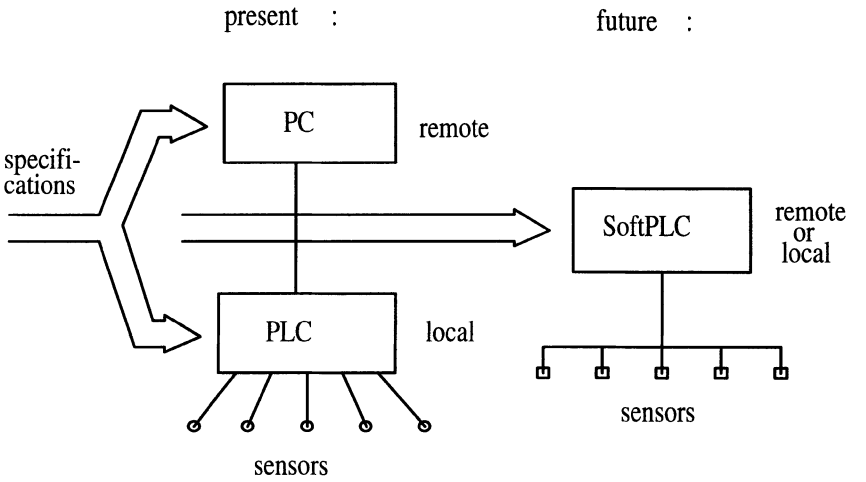

Figure 9 Future developments related to PC and PLC-programming.

9 APPLICATION TO DISCRETE PARTS MANUFACTURING (DPM)

The software packages for SCADA applications are attractive for DPM because they represent a large set of features also requested for discrete parts manufacturing. Additionally the packages are in widespread use for control tasks in the process oriented industries.

Two specific issues come up in this regard. One is the question of communication with CNCs, AGVs and Robots. A list of general communication requirements for this area is shown in table 1. The SCADA products can probably handle the communication requirements via device specific drivers written by experienced C-language programmers for the non standard manufacturing equipment at hand. The main issue is whether the SCADA-products can support the requirements for the manufacturing of discrete parts. In DPM there is a need, not only to monitor and control the state of the plant, but additionally to keep track of each individual part. This requires the SCADA product to be able to handle several plans simultaneously, one for each part. These, and other specific requirements that could make SCADA products much more interesting for flexible DPM are a topic for future investigations.

1) upload and download of programs to a device
2) dynamic downloading
3) notify supervisors
4) start and stop of programs
5) ask for status
6) change data on a device while executing a program
7) control auxiliary equipment
Table 1: Communication functionality for DPM -equipment (G.K. Christensen,1992)

10 REFERENCES

Fix Dmacs ver.5.0, System Setup, System Development, Advanced Tools and Display Development, Intellution, Inc. 1992-1994.

InTouch, ver.5.0 -manuals, Wonderware, Inc. 1995

MainStream- Application Integration Platform for Open Factory Systems, ITP Enterprice Software Inc, Cambridge, Ma, 1989.

SattCon 05 Slimline (V3) - Installation and maintenance, Alfa Laval Automation A/S, may 1994, Dok. nr. 493-0573-01 version 1.1.

DOX-5 Users manual, SattControl, AB Malmo, november 1990., Art nr. 493-0235-11.

B.Goran og G.Sandberg: "Funktionsdiagram - et kursushæfte til beskrivelse af SS IEC 848, S -konsult AB, ISBN 91-87-18220-3.

J.H. Christensen: "Function Block Standardization - Liason Report, ISO/IEC, Presented to ISO TC184 Meeting, Torino, Italy, 17/5 1995.

IEC 1131-3 (PLC Languages) IEC 1992.

G. K. Christensen and C. Nøkleby: "Quality Interfaces In The 90ties Using MMS", Systec, MAP/MMS workshop 3, 1992, Munich, Germany.

Rudolf Kanakovsky and Peter Woitzik: "Automatisierung der Typprufung von Software fur Prozessleitsysteme", Automatisierungstechnische Praxis 36 (1994). 10 , 12-21