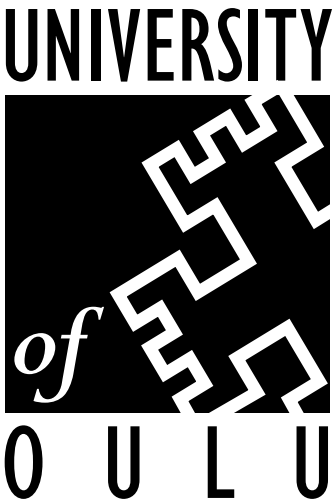

CONTROL ENGINEERING LABORATORY

DEFECT LOCALIZATION ON A PCB WITH FUNCTIONAL TESTING

Sébastien Gebus, Sébastien Lorillard,
Esko Juuso

Report A No 20, May 2002

University of Oulu
Department of Process and Environmental Engineering
Control Engineering Laboratory
Report A No 20, May 2002

DEFECT LOCALIZATION ON A PCB WITH FUNCTIONAL TESTING

Sébastien Gebus, Sébastien Lorillard, Esko Juuso

Control Engineering Laboratory,
Department of Process and Environmental Engineering,
University of Oulu
Linnanmaa, FIN-90014 University of Oulu, Finland

Abstract: This paper describes how Linguistic Equations, an intelligent method derived from Fuzzy Algorithms, have been used in a decision-helping tool adapted to the specific needs of electronics manufacturing. In our case the company involved in the project, PKC Group, is mainly producing control cards for the telecommunication and automotive industry. In their business, nearly 70 percent of the cost of a product is material cost. Detecting defects and repairing the Printed Circuit Boards is therefore a necessity.

With an ever increasing complexity of the products, defects are very likely to occur, no matter how much attention is put into their prevention. The work focused therefore on defect detection during the final testing of the product. The approach is based on experience using intelligent methods such as Fuzzy Logic or Linguistic Equations in fault diagnosis. An intelligent system based on expert knowledge was developed for analyzing test data. This analysis emphasizes localization of the defective components more than possible causes of those defects. Expert knowledge was essential for the development of the system as the number of defects is too low for a data-based approach. According to the first results, the system is successful for new products, even in the ramp-up stage. On the other hand, the underlying methodology provides techniques for tuning the tool parameters when amount of testing data increases. Diagnosis effectiveness can therefore be improved from detection of a functional area towards component level analysis.

This report is engineer oriented, which means that it has to stay simple and accessible to people without research background. In the first part of this report, we will give a short description of the different methods used in the project with exemples from the case study. The second part concentrates on what has been implemented in the company, what were the needs and what are the improvements that are expected from this work.

Keywords: fuzzy logic, linguistic equation, defect localisation, printed circuit board.

ISBN 951-42-6731-1
ISSN 1238-9404
ISBN 951-42-7513-6 (PDF)

University of Oulu
Control Engineering Laboratory
PL 4300
FIN-90014 University of Oulu

Table of Contents

1. INTRODUCTION

Production lines in electronics manufacturing are rather standard in their design [Gebus, 2000]. They consist in paste printing, component placement and soldering. They are very linear and sequential. It would therefore be quite easy to control the boards between each sequence, but in practice this is not done because of scarce resources. In the world of electronic manufacturing, the final product has usually only two different states: "working" or "not working"! The state "working" depends on the components used to make the Printed Circuit Board (PCB). We can of course also assume that each component has to fulfill a specific function (excepted if the designer was very tired and added some components in a decorative purpose). But what seems so obvious for most of the people has in fact a very important consequence. It means that the state "working" depends on every component and even the smallest defect on one single component can have a significant impact on the overall performance of a PCB. In other words, the electronic manufacturing world is a world where zero defects is a necessity.

Production with zero defects is, as we all know, an impossible target, and no production system can, and probably never will, achieve it. In fact, PCBs are nowadays getting more and more complex. From a purely statistical point of view, by increasing the amount of components on a PCB, very often you also drastically increase the defect rate. If you have for exemple a PCB made out of 1000 components, with a yield of 0.999 for each one of them, the yield for the PCB will be 0.999^{1000}=37%. It means that two thirds of the production would need some rework.

Fortunatly modern production systems allow us to increase the individual yields so that the global picture looks a little bit nicer. That is, by the way, the reason why ppm levels have been introduced already a long time ago in other industrial areas, where the biggest part of the job is to assemble spare parts without loosing in overall quality (automotive, aerospace…). It is of course always better to tackle a problem by its causes, however predictive and preventive actions on the production line reach at some point their limits. But since components cannot be repaired, is the understanding of a malfunction really the major concern in electronics manufacturing? Failure analysis provides many other challenges [Wagner, 2001].

We have therefore been looking at the production line from the point of view of the final testing. Because in-circuit tests become more and more complicated to implement, PCBs are going through functional testing. This is already common procedure in fault detection at chip level [Stout, 1998]. They have to be repaired if a defect has been detected. What we have tried to develop is a decision-helping tool using on one side intelligent methods for analysing the functional testing data and providing on the other side the information needed by the operator for repairing the PCB. This means mainly the localisation of the defective component.

Model-based approach has been an essential part of the fault detection already a long time [Isermann, 1984]. As the classical approach utilising analytical models has difficulties in handling uncertainties, fuzzy logic was included [Isermann, 1993]. The

results from dynamic systems were extended to other types of fault diagnosis [Juuso, 1994; Ulieru, 1993]. Fuzzy rule-based approach and linguistic equations were also applied to functional testing [Komulainen et al., 1997]. The overall Model-Based Diagnostical Process Analysis (MDPA) introduced in [Juuso, 1997] has been earlier extended to a Case-Based Reasoning (CBR) –type approach in a paper machine application [Juuso et al., 1998].

Nowadays, computational intelligence is becoming more and more important in fault diagnosis. As the linguistic equation (LE) approach is an efficient method for combining expertise and data-based modelling [Juuso, 1999], and there is long experience in using fuzzy logic in fault diagnosis, the fuzzy expert systems and their LE extensions were chosen to be the methodological basis of this project.

The goal of the project was to implement the functional testing tool to be used in production line. Architecture of the decision helping tool, and its connections to many existing systems must be analysed carefully. There are various methods for this analysis: the functional aspects of the system are represented by the SADT specification method, databases are commonly specified with E/R models, and the resulting systems can be described by flow charts. Finally, software tools for implementation must be selected so that the new system fits to the existing testing system. Efficient database management is essential, e.g. SQL is available in most database software. Internet and intranet application used by web browsers is feasible solution for integrating different requirements of the decision helping tool.

This report consists of two parts. In the first part, methodologies of computational intelligence suitable for diagnostical applications are described (section 2) and applied to the testing problem (section 3). The second part describes how in practice a Decision-Helping Tool based on Linguistic Equations has been specified (sections 4 and 5) and implemented (sections 6 and 7) for PKC Group. Finally, some results from the production line are shortly presented.

2. COMPUTATIONAL INTELLIGENCE FOR DIAGNOSTICAL SYSTEMS

"Electronic Diagnosis is the process of identifying the components or connections that are responsible for the malfubction of a defective printed circuit board so that corrective action can be taken both to repair the board and to improve the process" [Balakrishnan, 1999]. This system has to help operators to repair PCBs. It has to be simple to use by both designers and operators. Currently operators only have their own personal experience when they have to make reparations on PCBs. Unfortunately this experience isn't available in a way that could benefit to new employees, nor is it of any use in the case of new products.

2.1 What is an Expert System?

An Expert System is a program having a huge amount of knowledge about a specific topic. By gathering this knowledge obtained from experts, we hope to create a system as efficient as those experts.

This knowledge can be divided into two forms:

- Knowledge about the environment, facts that are known to be true. In this category, we put for example the functional testing or information about the different defects.

- Knowledge about the interactions between the previous facts. This part is the engine of our system, it contains the rules and it takes the decisions.

Figure 1 represents how an Expert System is working. You can clearly see the two kind of knowledge, with the rules making the link between functional testing data and defect information.

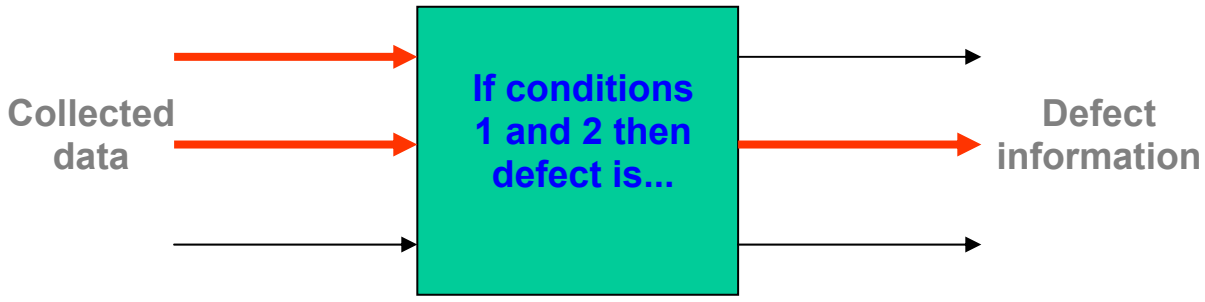

Figure 1: General representation of a rule-based system

2.2 How to build an Expert System?

2.2.1 The hard task of finding the “experts”

For several reasons it is indeed not an easy task to collect the necessary expertise.

The first reason is that the knowledge that is needed isn’t the property of one single individual. Very often a high level of collaboration is needed between several departments, which are not used to work or sometimes even communicate together.

Then, there is a reason inherent to the definition of what is an expert. An expert wouldn’t be one if his knowledge would be so easy to get. When you are trying to get someone’s experience, don’t expect him to give you the formula that will solve all your problems. Expertise is something very subjective and it is therefore necessary to submit your rules to the expert’s approval. It seems obvious, but asking the right questions can also help to get the right answers. But anyway, translating the expertise into a model easy to run on a computer is very often a cause of headache.

Finally, experts are usually very busy people and if you tell them that they are experts and that you need them, they tend to become even busier. What we try to underline here is that if you are not enough of a psychologist, you will probably only manage to scare them away!

For all those reasons, getting expertise and putting it into something that can be used with computers can be a long and difficult task.

2.2.2 The development process of our Expert System

Figure 2 shows how in our case we have defined the specifications for our Expert System.

The environment was not difficult to define since we were in a situation where we could only get information from functional testing on one side and on the other side we wanted to focus on the localization of the defective components. So, even if we couldn’t clearly qualify the environment, we could at least define the limits.

Defining the rules was a much more difficult task. One solution would have been to analyse historical data and to try to find patterns in this data so that we could link them with specific defects. In an ideal case, it could even have been possible to make a model and to analyze the propagation of the effects of defects through that model [McKeon, 1989]. Unfortunately, despite the huge amount of data available, there was no link between any defects and the related measurements and this data was of no use. This underlines the incompleteness of many expert systems that eventually only work with very simple PCB architechture [Tong, 1988][Tong, 1989].

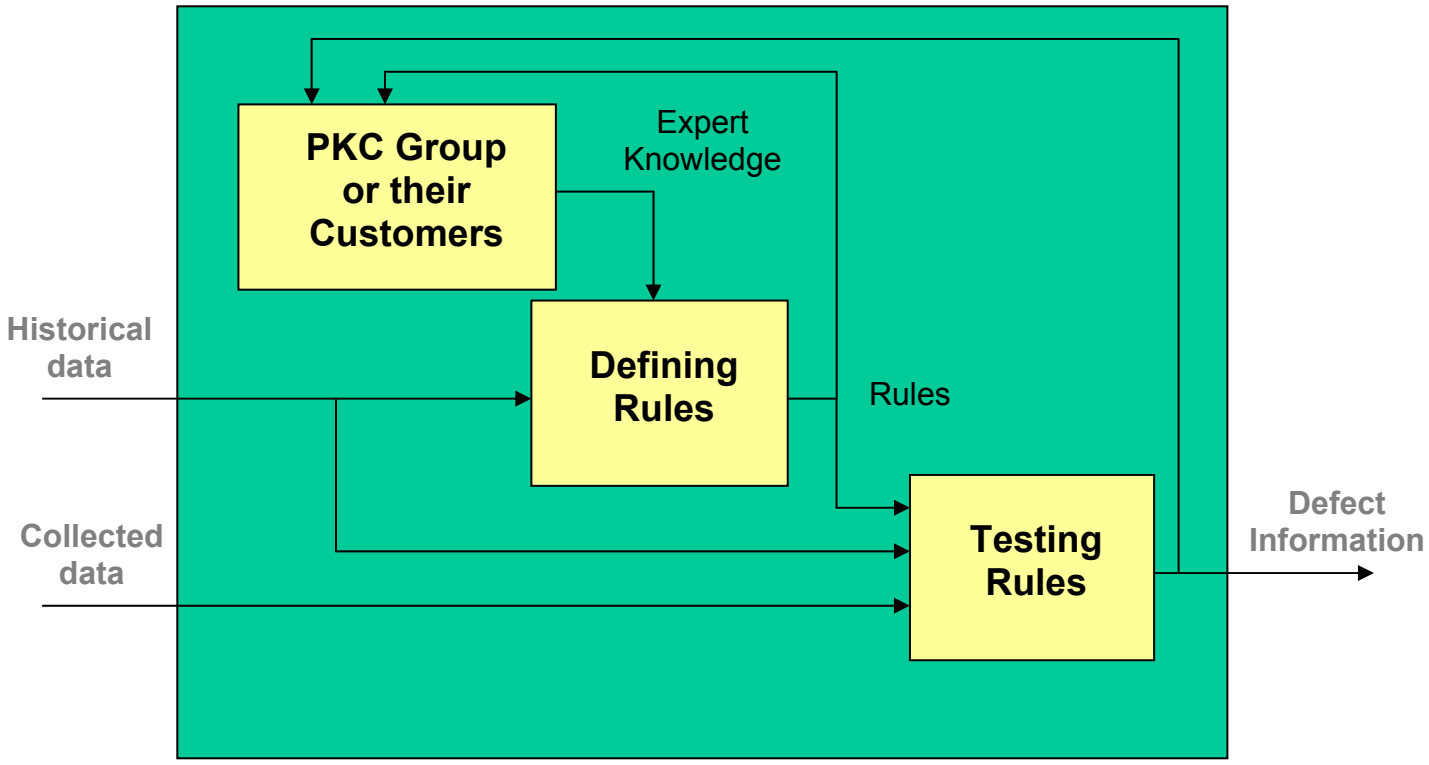

Figure 2: How to build the rule-based system?

Therefore, we had to use expert knowledge to generate a set of rules. That process took quite a long time and one of us had to be assigned at full time on the problem before it produced some results, but it was then amazing how much information we were suddenly able to retrieve. That information came from test designers and it was nothing less but the links between every single function that was tested and geographical areas on the PCB containing a small group of components.

We had rules for a rough localisation of the defects on the PCB, which means that the heart of the system was set!

2.3 Fuzzy Logic

Having rules is important because it explains us how the different parts of our environment are interacting with each other. We are now able to say things like for exemple: “When voltage V is low and current C is high then we have a problem”.

But it also means that we not only need to know what are the elements composing our environment, but also how we can qualify them. What does it mean that V is low or C is high? That’s where Fuzzy Logic is used.

2.3.1 The basic concepts

Fuzzy Logic is a method for dealing with uncertainty. Instead of using mathematics, it offers a rather pragmatic approach of problems. In fact, it is a way to represent human reflexion. Both input and output variables are qualitative (low, normal, high…) instead of quantitative.

Solving a diagnostical problem with fuzzy logic, usually includes the three following steps:

- Fuzzification of the variables, whether they are input or output variables
- Specification of a set of rules
- Combination of rules for generating outputs

Fuzzification of the variables:

Fuzzification is based on the fact that Boolean variables usually give a bad representation of reality. If we take the example of the low voltage, a Boolean representation would tell us that under 40V voltage is low and suddenly, at 40.01V voltage becomes high. A fuzzy representation allows us to have a smouth transition from low voltage to high voltage by introducing degrees of membership (Figure 3).

In the first case, the state “high voltage” starts at 40V and voltages under 40V are simply excluded.

In the second case, 38V means that the voltage is high with a degree of membership of around 40%.

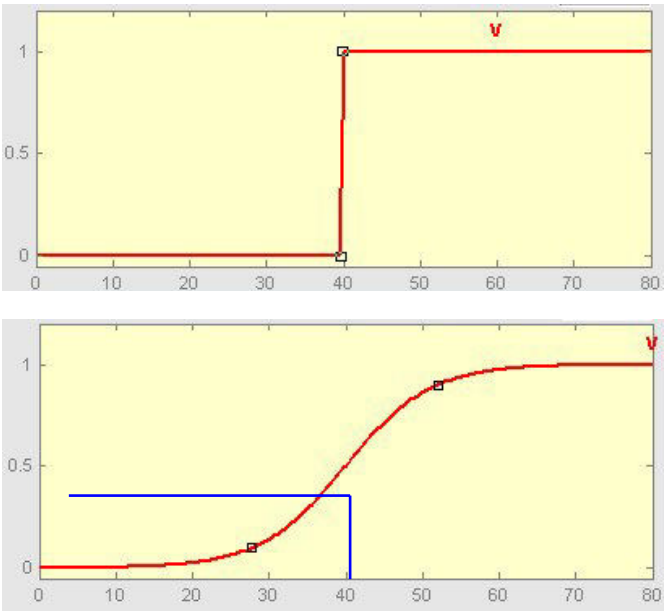

Figure 3: Boolean and Fuzzy representation of a variable

Figure 3 represents the fuzzification of one single state of one variable. Of course, we will most of the time have to represent different states. The result can then be seen on Figure 4. In this case, we will consider the degrees of memberships. 30V for exemple is “low” at 70%, “normal” at 25% and “high” at 5%.

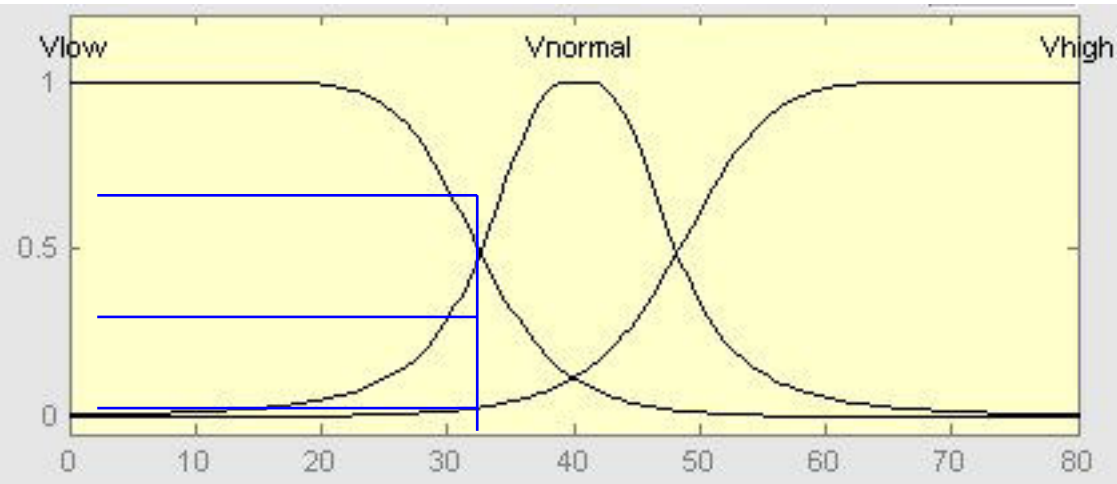

Figure 4: Degrees of membership

The fact not to have a crisp transition between the different states improves the stability of the system. Boolean systems become unstable when values are close to the limits.

In our case, expert knowledge gave us only Boolean limits. Fuzzification has been done at a rough guess, but the system is designed in such a way that it will tune itself as soon as additional data will be available.

Specification of a set of rules

After having fuzzified all the variables, we now have to define rules to link input to output variables. This link is also made with degrees of membership. It means that if an input is true with a degree of membership of 70%, the associated output will be true as well with the same degree of membership (Figure 5)

A method often used for defuzzifying the output is to take the center of gravity of that output.

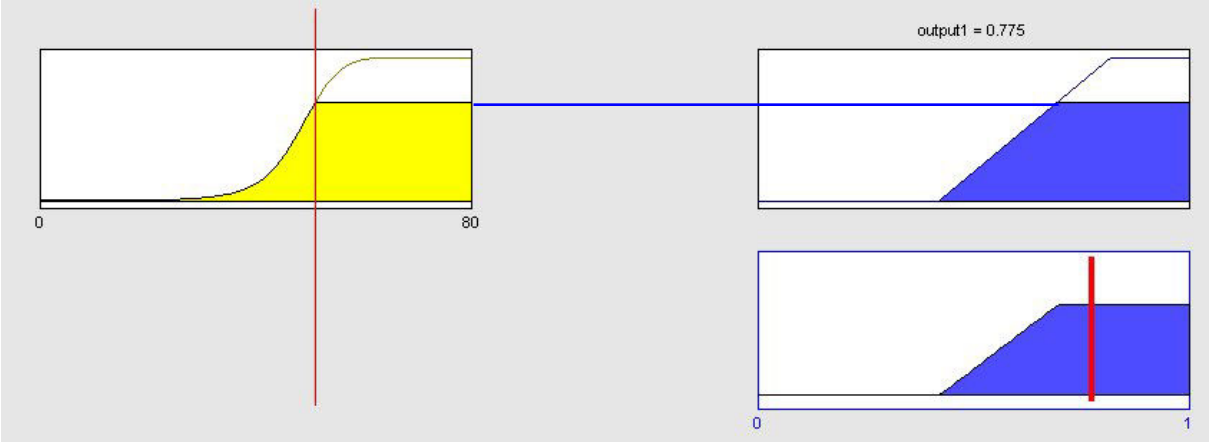

Figure 5: input/output interaction for one state of a variable

Combination of rules for generating outputs

In practice, rules are usually more complex than the previous exemple. Logical operator like “AND” or “OR” can be used to combine rules.

Combining the inputs is then most of the time done by using maximum (operator AND) or minimum (operator OR) degree of membership.

Combining the outputs is commonly done in the way explained earlier. In the case of more than one active output state, we are looking for the center of gravity of the resulting picture as shown on Figure 6.

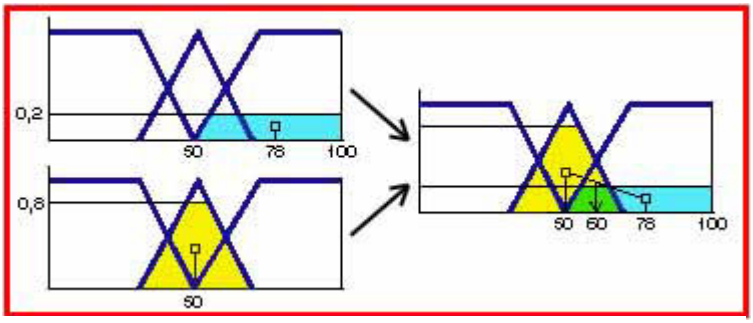

Figure 6: Output aggregation

2.3.2 Limitations of fuzzy logic toolbox

Like many ready-made generation systems, the Fuzzy Logic Toolbox available in Matlab¾ didn't provide us a suitable answer to our problem. The very attractive graphical interface makes it easy to configure simple systems with few inputs and few outputs without even having to write a single line of code. But in our case, the amount of input variables was too big to use the toolbox properly. It loses its interest as a testing tool if it is not flexible enough to test complex systems. In fact, several fuzzy systems could be combined in a Simulink system by using Fuzzy Logic Controller blocks for implementing specialised fuzzy inference systems. However, generating numerous specialised systems is a time-consuming task.

As an experimentation tool, Matlab¾ has some interesting points. It is possible to create for example different models and to test them with different parameters. However, we wouldn't go as far as saying that Matlab¾ is a solution. An experimentation tool provides information about experiments. It answers the question "What method can we use to solve this problem?" by checking the viability under different hypotheses. But it's not a tool like this that will run, monitor and take decisions on a process in a factory. The development of the solution is therefore very often independent of the tools that have been used to specify that solution.

In the project with PKC Group, we had the advantage to be able to develop the solution on-line. By working in a real environment, it was possible to assess also the needs from the user point of view. It is sadly very often forgotten that a method is only part of the solution. Formulas or an algorithm only represent the researcher point of view, but in order to be effective they have to be associated with an interface. It is as much part of the solution as the method itself. That's one reason why the method used in this project is based on Linguistic Equations. Close to fuzzy logic, they are easier to implement and only limited knowledge is needed to use them.

2.4 Linguistic Equations

Linguistic Equations is a relatively new method, developed by Esko Juuso at the University of Oulu [Juuso, 1999]. The general idea behind Linguistic Equations makes this method in its approach very similar to Fuzzy Logic. Where we had previously fuzzyfication – rules – defuzzyfication, we have now "linguistification" – equations – "delinguistification". Linguistic equation systems can be developed with FuzzEqu Toolbox implemented in Matlab¾ [Juuso, 2000].

2.4.1 Linguistic variables

The first analogy with Fuzzy Logic is the form of the data. Before using it, the data is put into a more explicite way by applying membership definitions. Like shown on Figure 7, the resulting variable is a new value between –2 and 2. This value represents the level of the variable, from very low (-2) to very high (+2).

With this representation, we suppress some problems linked to the implementation of the degree of membership. A value "normal" at 80% and "high" at 20% will in this case have a linguistic value of 0.95 for example. In practice, it is easier to handle a value than several labels with different degrees of membership.

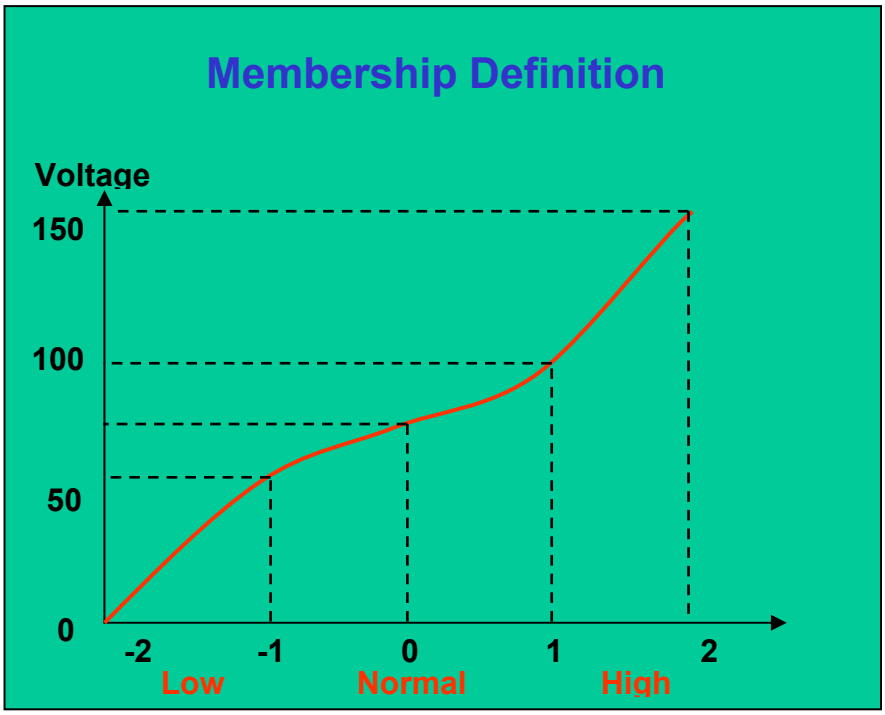

Figure 7: Membership Definition used to obtain Linguistic Variables

Membership definitions are used for non-linear scaling of the data. A membership definition is a bijective function between measurements and a Linguistic Variable. The shape of that curve represents not only the different states (low, normal, high…), but also the variation of the measurements. It is meant to reduce non-linearities in the measurements. We therefore try to find linear interactions between linguistic variables. This property is used when generating the Linguistic Equations.

In practice, the membership definition is obtained by polynomial regressions after having calculated the values for the measurement relative to the five main values for the Linguistic Variable. Two second order polynomes are then used, one for the interval [-2 0] and the other one for the interval [0 2] with respect of the strict monotony. In theory, other forms of regressions could be used, but in practice, second order polynomes give a good representation of what is the variation of the measurements in a real process.

In the FuzzEqu Toolbox, membership definitions can be either analysed from data or defined on the basis of expert knowledge.

2.4.2 Generating the Linguistic Equations

In the same way that Fuzzy Logic is generating rules based on Fuzzy Variables, a Linguistic Model uses Linguistic Variables in order to generate equations. As said earlier, the idea behind using Linguistic Variables is to obtain a linear system. Linguistic Equations are therefore also linear equations.

From an easy case: 1 input, 1 output…

Let us take an example that illustrates this property. In this very simple case, there is only one input X and one output Y. We also suppose that those two variables are related to each other because it doesn't make sense to analyse a randomly distributed data. Figure 8 shows what happens to the data when membership definitions are applied.

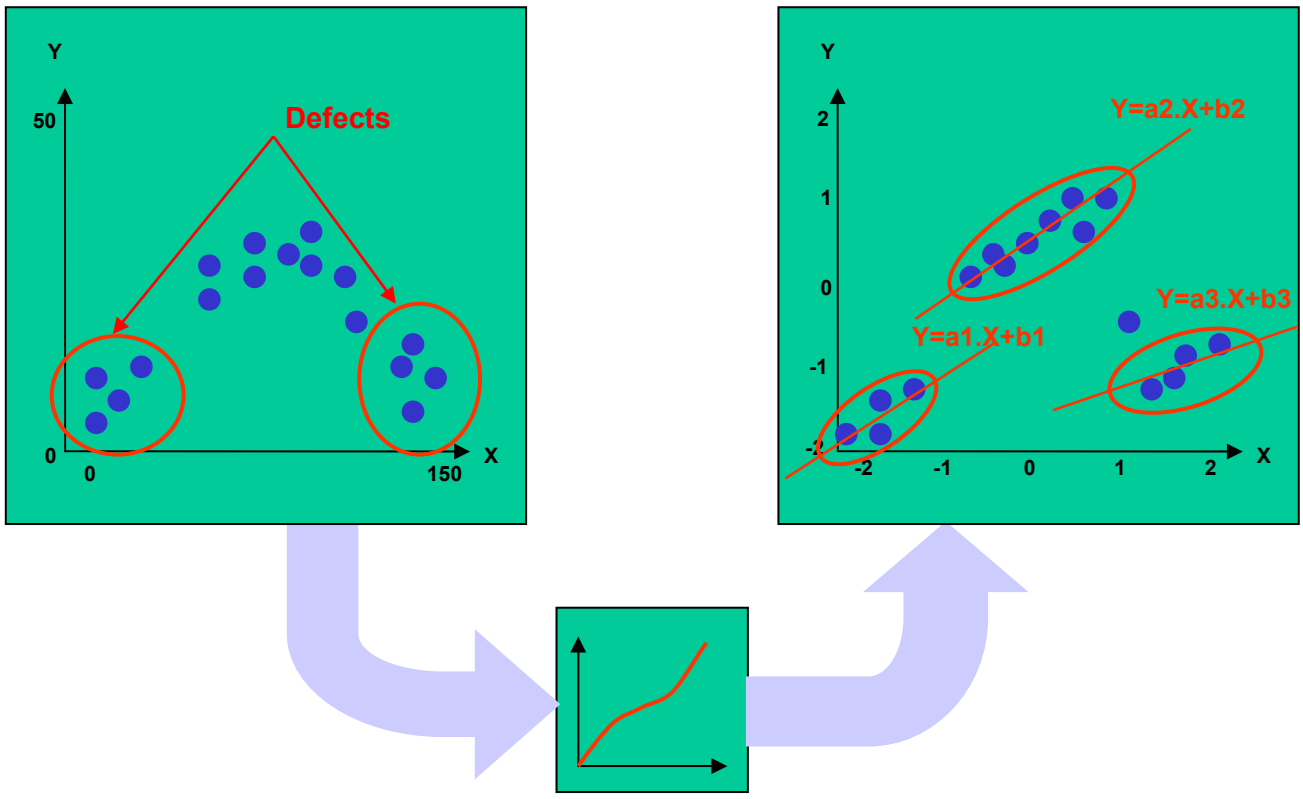

Figure 8: Linguistification of the variables and Linguistic Equations

Fuzzy Logic would take the measurements in their original form and sort them into different categories. The two defect situations would be handeld with fuzzy rules:

- If X is “normal”, then Y is normal
- If X is “low” or “high”, then Y is low

But a linguistic model is looking for linear relations between linguistic variables. By applying clustering methods and linear regression to this data [Hyötyniemi, 2001], we obtain equations for 3 lines. One line $*Y ? a_2 X - b_2 +$ represents the normal situation and the two other lines represent defects:

- If $Y / a_2 X / b_2 ? 0$, then situation is normal
- If $Y / a_1 X / b_1 ? 0$ or $Y / a_3 X / b_3 ? 0$, then there is a defect

Instead of applying rules, we only have to check how close our measurements are to those lines. The ideal situation is when those lines are far apart from each other and never cut each other over the interval [-2 2]. If we have sufficient knowledge about the defects, it is also possible to define a different equation for every one of them.

...To the general case: n variables

What is true for one input and one output can be generalised to a large number of variables, for example if the aim is to study the influence of a group of input variables on one or several outputs. In theory, the only difference is the use of multilinear regression instead of standard linear regression.

$$Y ? a\ X - b \qquad \text{becomes} \qquad a_1 X_1 - \ldots - a_n X_n ? 0$$

Depending on the measurements that we chose, we obtain different vectors (a1,…,an) of solutions to the previous equation.

$$a_1 X_1 - \ldots - a_n X_n ? 0 \quad \Rightarrow \quad \text{no defect}$$

$$b_1 X_1 - \ldots - b_n X_n ? 0 \quad \Rightarrow \quad \text{defect 1}$$

$$\ldots \quad \Rightarrow \quad \ldots$$

In practice however and especially if there is a large amount of variables, many of these coefficients are expected to be close to zero.

One reason for this is that by increasing the amount of variables (because of a lack of knowledge about the model for example), you also increase the risk of including variables that simply don't have any effect on the output. By a mere chance, it could be possible to find some linear relation between one of those variables and an output but in a general case, they will just produce some random noise in the model and be eliminated by the multilinear regression.

Another reason is that very often there is redundancy in the parameters that are being analysed and some variables don't have a direct effect on the output. For example, an output that depends on 20 different inputs could in fact only depend on 5 of those inputs, each one of them being a result of the 15 others.

In practice, good equations should eventually not contain more than 5 or 6 variables. If it is not the case, the system will show redundancy in the parameters that are being analyzed and some variables don't have a direct effect on the output. More variables would therefore mean that the problem is over fitted.

Each equation can be considered as a case-based model in the Case-Based Reasoning approach presented in [Juuso et al., 1998]. The decision is based on how much the left hand side of the equation differs from zero when the scaled measurement values are used, i.e. how close to the surface the measurement point is. The difference defines the degree of membership for the case. More generally, each case can have also more than one equation model, i.e. all the equations which should be true at the same time. Then the conclusions must be aggregated from the degrees of membership calculated for individual equation models.

3. CASE STUDY: PKC GROUP

The problem, as it appears in PKC Group, is slightly different to the general case presented above. Originally, a data-based approach was planned to be used. However, the number of fault cases was too low for data-based analysis; also the fault information was not easy to connect to the test measurements. Several modifications have been made to the method in order to adapt it to this particular case.

3.1 Variables

Inputs and outputs

In our case, outputs are geographical areas on a PCB. They are therefore not variables and equations will only contain inputs (measurements). A link exists between an equation and an area, but an area cannot be part of an equation.

$$a_1 X_1 - \ldots - a_n X_n \; ? \; 0 \qquad \textbf{with } {}^{*}X_1,\ldots,X_n{}^{+} \textbf{ only inputs}$$

Weight of each input

For each area, input variables are known to be independent and have an equivalent effect on the status of the area.

$${}^{*}a_1,\ldots,a_n{}^{+} \; ? \; {}^{*}1,\ldots,1{}^{+} \quad \textbf{and the equation is simplified} \quad X_1 - \ldots - X_n \; ? \; 0$$

3.2 Definition of the limits

For each variable, if the measurement crosses a certain control limit, it means that the area has a defect. Since the control limits are known and well defined, it makes sense to assign -1 and +1 as a linguistic variable to those limits. As for the central point, 0 is the linguistic value that represents the average of the measurements. By making those choices, we can obtain membership definitions that respect the repartition of the measurements between an average value and a control limit. The shape of the membership definition can give us a lot of information about the variable.

Figure 9 shows the membership definition for a variable with the following properties:

- Measurements are shifted towards the lower control limit
- A “low voltage” will have a much bigger effect on the linguistic variable than a “high voltage” and will therefore be detected much faster

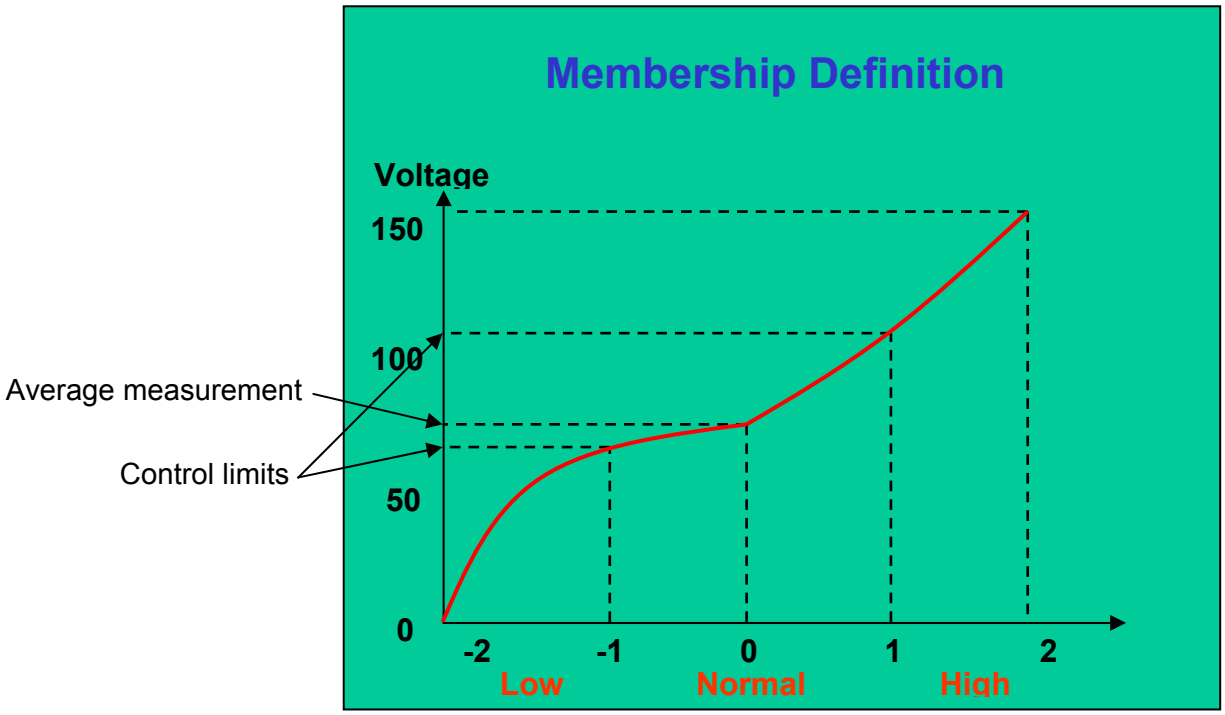

Figure 9: Membership Definition when measurements are not centered

By using this representation, we also emphasise the fact that a system that is under control but not centered is better than the one, which would be centered but has a high variance. The first kind of system will always have linguistic variables close to zero.

3.3 Linguistic Inequations

According to what has been said, for each area we are now able to state two general rules:

If every variable is "close" to zero, then the area is OK,

Whatever variable we consider, we never have $|X_i| \,@\, 1$.

By applying those two rules to the equation and including the standard deviation on the measurements, what we obtain is not anymore an equation but an inequation that has to be verified in order to have an area without defects:

$$|X_1| - \ldots - |X_n| > 1 - u \qquad \textbf{with } u \textbf{ the deviation}$$

As it has been explained in the general case, the aim was not to find equations, but rather to check how close our measurements are to those equations. It means that anyway we would have eventually ended up by checking inequations.

The deviation has been calculated by

$$u ? k.\hat{A} u_i / \overline{u}$$

where u_i is the standard deviation for the linguistic variable X_i and $\overline{u}$ the average standard deviation for the linguistic variables.

Coefficient k is a corrective factor that depends mainly on the amount of available data and the diversity of the variables. A good case would be to have a lot of data for many variables with a similar standard deviation, but unfortunately that never happens. This means that coefficient k can be used to tune the system in case it detects inexistent defects for example.

The final form for our inequation is then:

$$X_1 - \ldots - X_n > 1 - k\frac{n/1}{n}\hat{A} u_i$$

However, it is regrettable that we don't have any defect information. In a general case, there would be different equations with different coefficients for each defect type. Since we don't have that information, we are only able to draw a line in our n-dimensional space and to say that all the points that appear to be close to this line are good. But we are unable to say where is going through the line that would for example represent a short circuit or a broken component.

3.4 Advantages and disadvantages of the method

The first and probably major interest in the method is probably how easy it is to implement. Membership definitions are calculated with a polynomial regression on three points (two of them are known and the last one is the measurement average) and the equations can be defined in advance. Once this has been done, implementing the system consists in two steps. First step is to apply a function that represents the membership definition to the measurement data in order to obtain Linguistic Variables. Then applying those Linguistic Variables to an inequation and checking the result. This process is fast and can be used on-line as a real time system.

The method is also reliable after a very short ramp-up period. Tests have been made with new products and membership definitions have been calculated with the data from 15 prototype PCBs. In this case, the system had 90% of good results. We consider therefore that the system can be reliable enough even after one day of production. However, it is possible and advised to tune the system when more data is available

The robustness has also been checked during our tests. Besides the fact that it works pretty well even with new products, we checked the results with products that are known to have problems and others with a very high yield. By adjusting the corrective factor, it was always possible to get good results. Same comment can be made about the effect of different kind of measurements. Both analog and binary measurements could be used.

The system is configured with almost only parameters defined by “experts”. It becomes therefore very simple to tune it, since only few parameters can be modified. Unfortunately this last point is probably also the biggest limitation to the system. By not relying enough on experimental data, we only get an approximative knowledge. As it has been already explained, it is for example impossible to identify defects from each other. Even the localization of that defect on the PCB is limited to an area composed of 10 to 20 components. However, it is an improvement if we consider that there can be hundreds of components on a PCB.

4. THE DECISION-HELPING TOOL

4.1 The production line and the INTELE System

Currently the production line can be represented as a succesion of machines and humans, placing and soldering components on PCBs (printed circuit boards). At the end of the production line, a test station is measuring voltages, currents and times at specific checkpoints. This functional testing tells us if the board is working or not. In case a problem is detected, the board goes to the repair area where operators have the laborious task of finding the defect. Measurements are the only information that they have for this task.

Their work is therefore only based on their knowledge of the product and experience of former defects. Considering that we do not have mass production, this experience stays rather limited. As a consequence, it takes a lot of time for repair-operators to find defects.

We can consider a production line as 4 entities (Figure10):

- Production: This entity that makes PCBs (machines and operators).
- Test station: The entity that tests the functions (test station + an operator).
- Reparation: The PCB is being repaired if a defect has been detected (operator).
- Quality department: Quality department is collecting defect information and provides feedback to improve the process and the production.

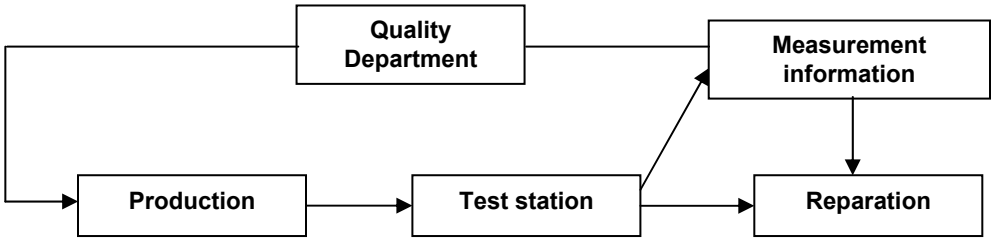

Figure 10: Production line.

So we could say that currently everything is based on operator's knowledge and experience. But such a situation is not a reliable one because people always do make mistakes. Then if an operator leaves the company, his knowledge and experience will be lost and investments will have to be made in order to train a new one.

What is therefore needed is a Decision-Helping Tool (Figure 11) that analyses measured data and gives localization and some kind of information about the defects. We can then have a reliable system, increasing the flexibility of the production line. The only necessary knowledge for operators is to know how to use the software. Furthermore, besides containing all the knowledge about the products, the system will also be a real source of information for the company.

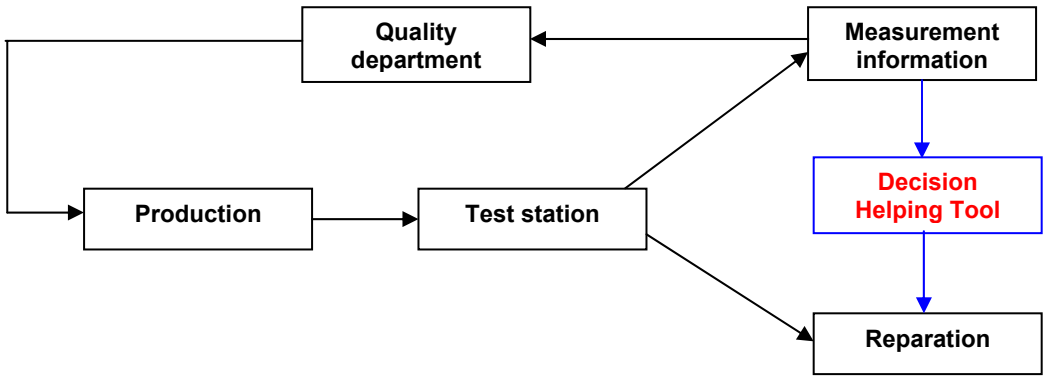

Figure 11: Production line with the decision-helping tool.

4.2 How does it work?

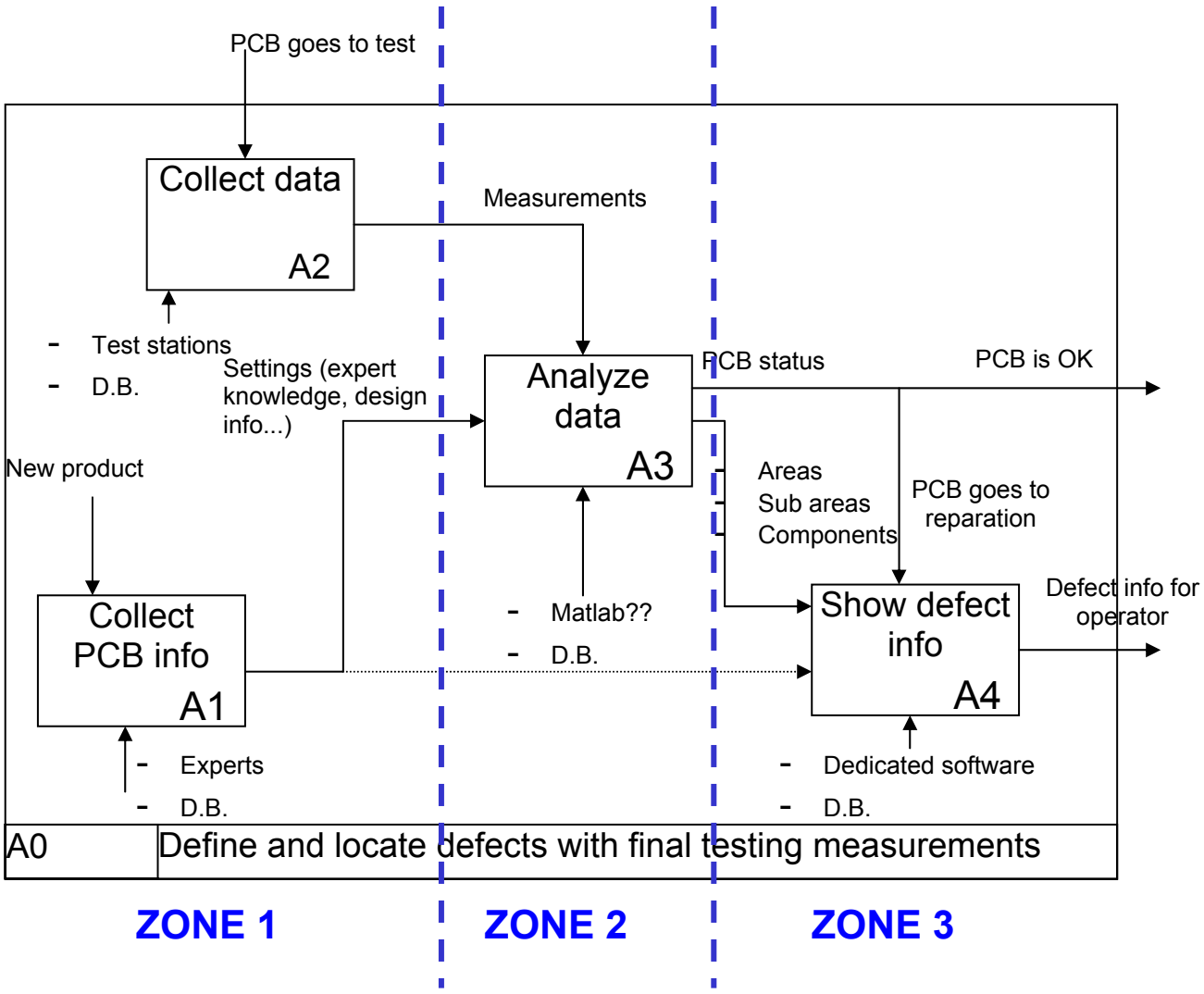

Figure 12: Global definition of the Decision-helping tool.

This description (Figure 12) is obtained by using the SADT specification approach. This way, it is possible to represent the system from a functional point of view. It shows the interactions within the system but also between the system and the rest of the process making the implementation work easier.

Zone 2 for example represents the heart of the system where the rules and the linguistic equations are analyzing data. It's the intelligent part of the system, whereas the first zone's function is to collect the company's knowledge. There are two ways to collect the knowledge, from experts or through extensive data collection. Zone 3 being the interface, its function is to show defect information in the most appropriate way.

We have now clustered our initial problem “detect and localize defects for the repair operators” into three smaller ones:

- Collect the needed data
- Transform it into information
- Get this information back to the operator

It is possible to deal with each of those problems separately, but by using SADT the separate solutions will also fit to the initial problem. In the same way that we have object-oriented programming, we have here a function-oriented system.

4.3 Architecture of the system

Since we are trying to build an intelligent system, why not explain its architecture by using an analogy with what is probably our best known intelligent system, the human being? Human architecture is working since now some centuriues very well.

Figure 13 and Figure 14 describe similarities between the Intele system and the human way of working and interacting with the environment. We can consider that our system is like an expert connected to the process.

This expert is first learning about the product (mapping of the components, grouping of those components into functional areas…). Several means of communication and the memory is used for this purpose. In our system, these means of communication are softwares used by desing-operators to enter a product definition and other useful information. Once a product has been defined, a patern is “memorized” in a specific database.

Now that our expert knows about the product, he has to apply this generic pattern on other products. The connection to the process is made by accessing measurement databases. By combining the data with the knowledge about the product, it is possible to provide defect information to the repair-operator.

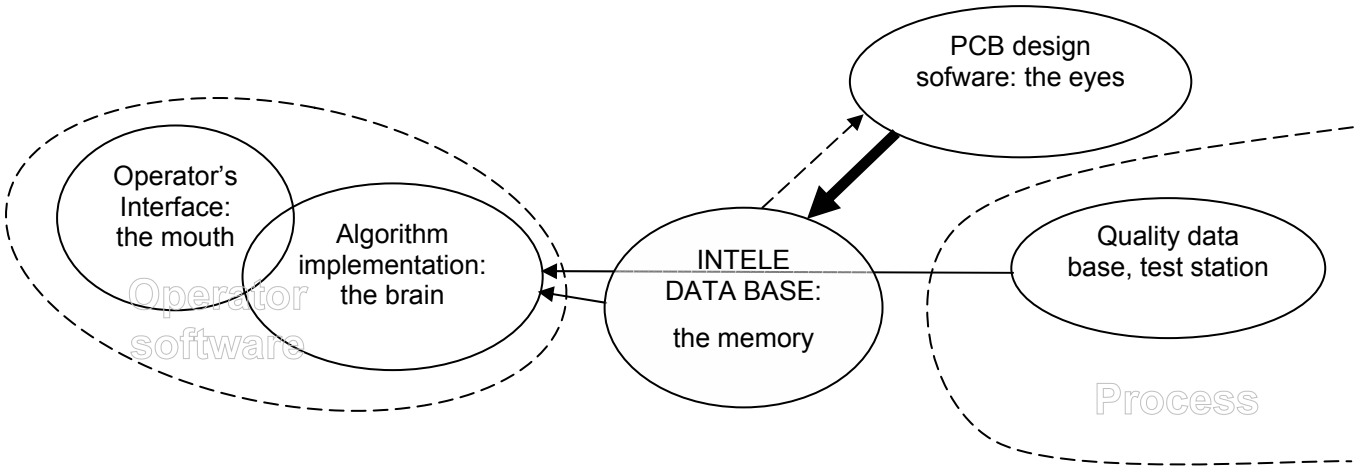

Figure 13: Physical architecture of the decision-helping tool

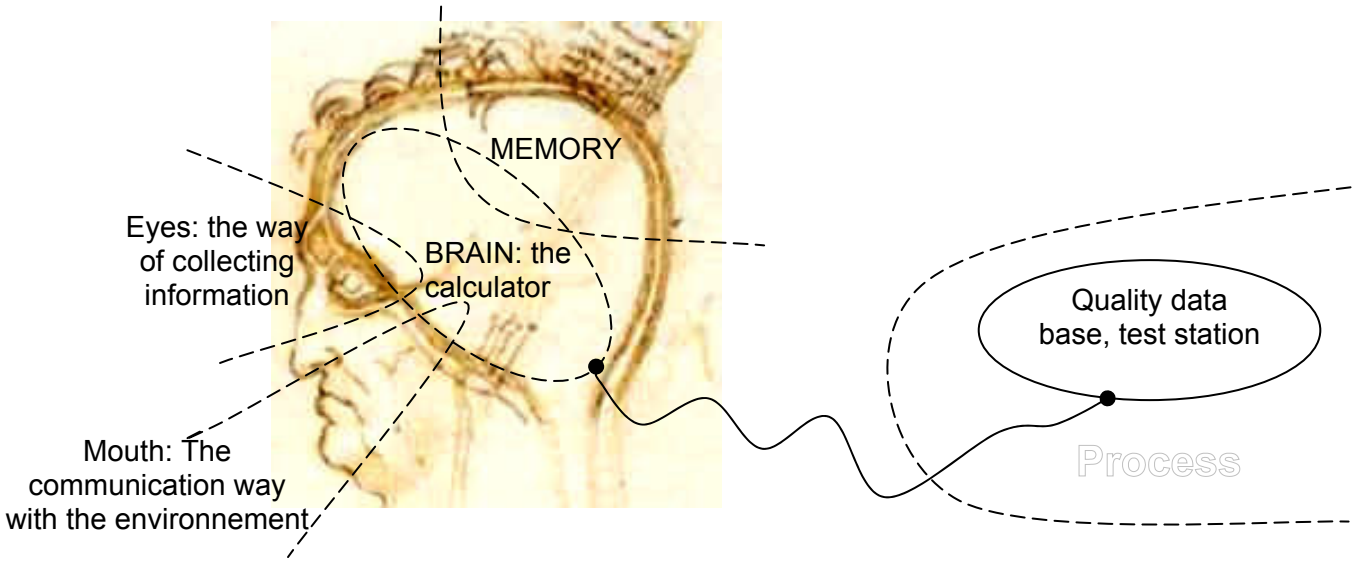

Figure 14: Human architecture.

5. A SPECIFICATION APPROACH WITH SADT, E/R MODELS AND FLOW CHARTS

In all projects, a clear definition of the targets is a key to success. We already presented shortly how SADT method could be used for a functional analysis and therefore a better understanding of a system. Actually three different specification methods have been used during this project, each one of them used for a specific need of the system. By using those methods, it is possible to define a precise frame for the project.

5.1 SADT for the general approach

SADT is generally used as a specification method during the design or re-design of a system through a global and functional approach. The main idea behind this method is to define a system by splitting it into sub-systems. Each one of those sub-systems is again splitted into even more basic sub-systems until a proper level of details has been reached. We can consider that the level of details is sufficient when we obtain sub-systems of such a basic level, that they describe a very simple function and its interactions with the rest of the system.

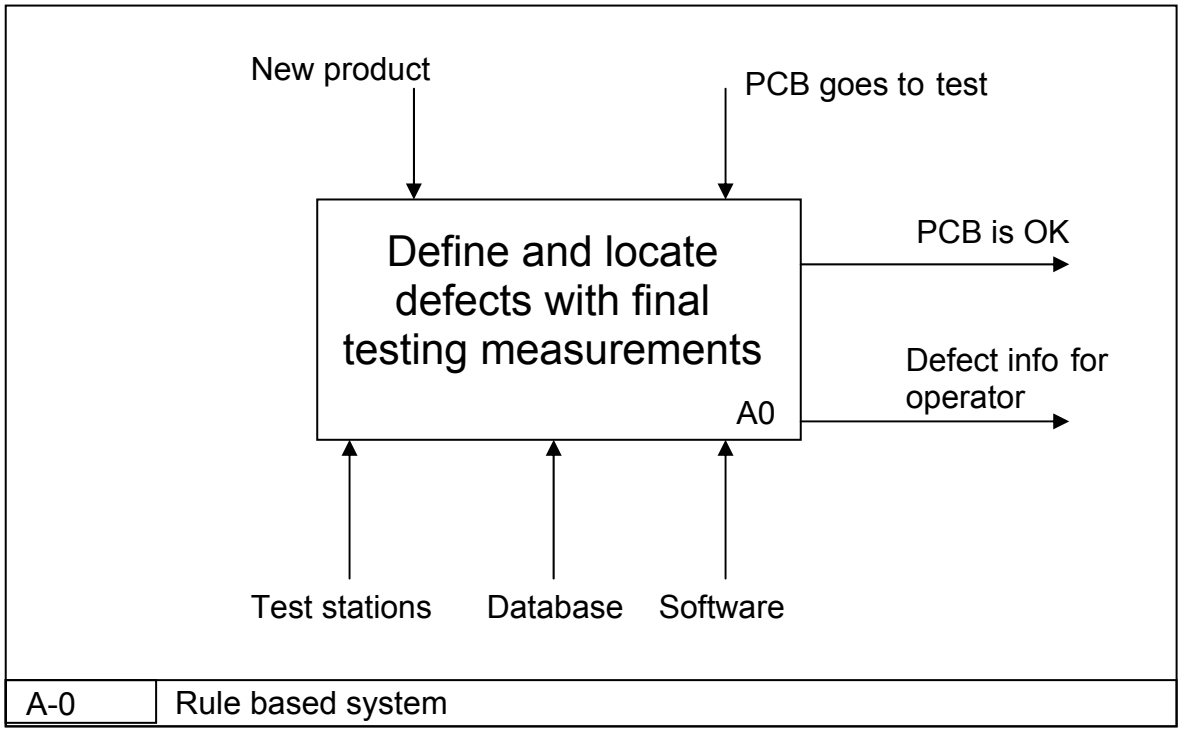

Figure 15: Decision-helping tool: global definition.

Figure 15 describes the first level of the method (level A-0). It is a global definition of what the system should do, based on some equipment (test stations, database…), events like the introduction of a new product, and producing some outputs (defect information). Figure 12 is the second level and gives already much more details especially about the interactions between some of the sub-systems. In this case we stopped the SADT approach at the third level, because the idea was to use this method only for a general purpose and to use more specific methods for other parts of the system.

5.2 E/R models for the database

Entity/Relations (E/R) models are the most common way of specifying a database. They are like maps of the database representing the links between the different tables. The fact that every database designer knows the method makes it even more important. Since everybody can understand it, it creates a common understanding and is a perfect base to discuss plans for database modifications for example.

Figure 16 specifies the whole Intele database. In this model each box represents one element. Each element has some properties like name, value, side, position. The circles represent links between two elements. Between “input variable” and “input state” for example, there is a belonging link with two limits, meaning that for each couple (“input state”, “input variable“) we have two specific limits. With this kind of model it is easy to represent the way the database will work. It also makes it easier to write the SQL requests for accessing the right data.

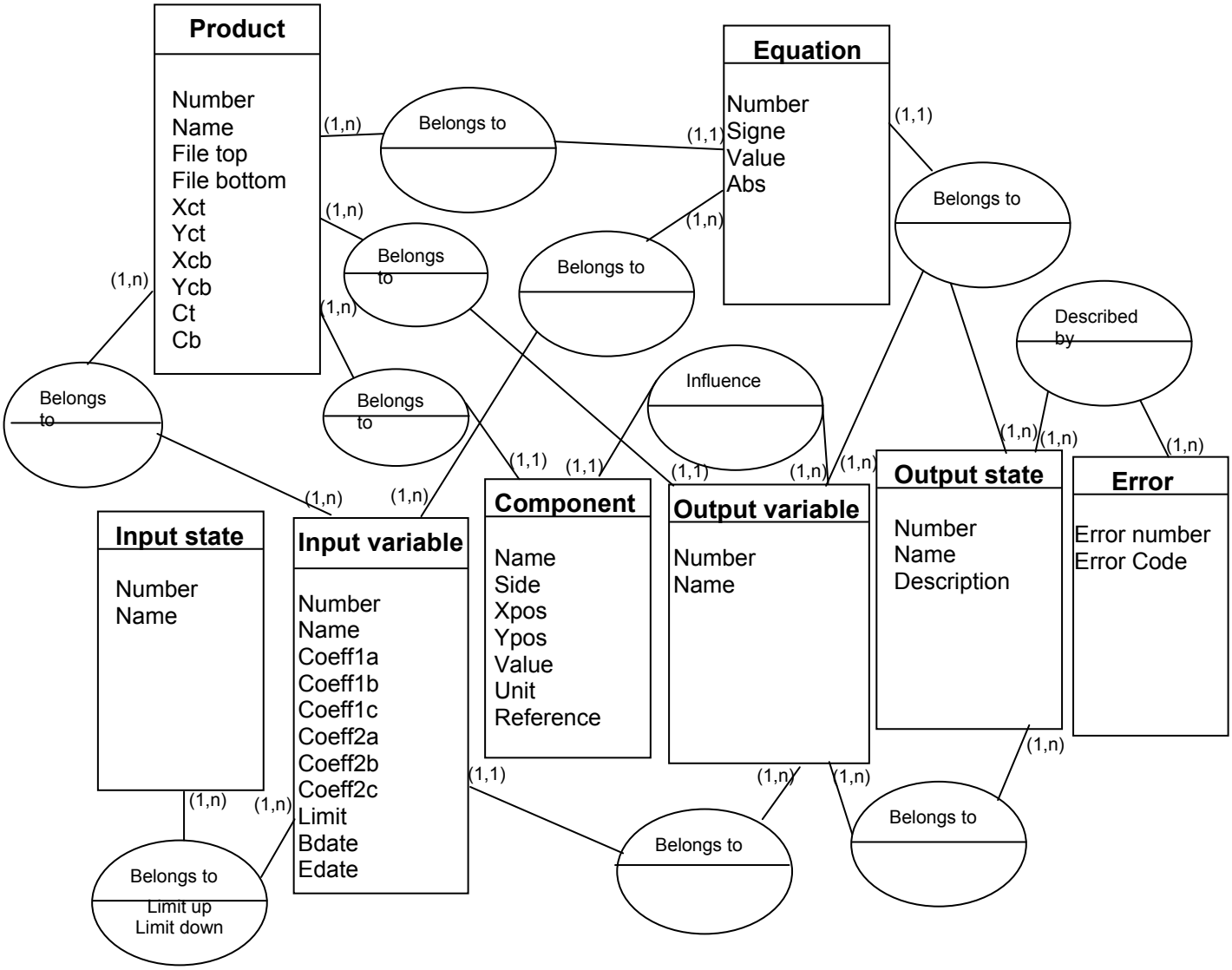

Figure 16: E/R model for the database.

5.3 Flow charts for software specifications

Flow charts are very easy to understand for everybody. Their ordonated structure makes them a rather natural tool to represent how the software is working. In our case, they have been used for the specification of the software for designers and the operators. By including the requests of the end users it was possible to design softwares that really meet the requirements of the people who will have to use them.

The use of flow charts is also interesting in the way that it creates a link between software designers and the "real world" surrounding them. Those people sometimes tend to forget that they not only work for the sake of writing lines of code, but that there is a real aim in doing so.

5.3.1 Operators software

This sofware is the brain and the mouth of the system. It analyzes data from the process and gives the operators information about the defects. This analysis is made with an algorithm based on linguistic equations (Figure 17).

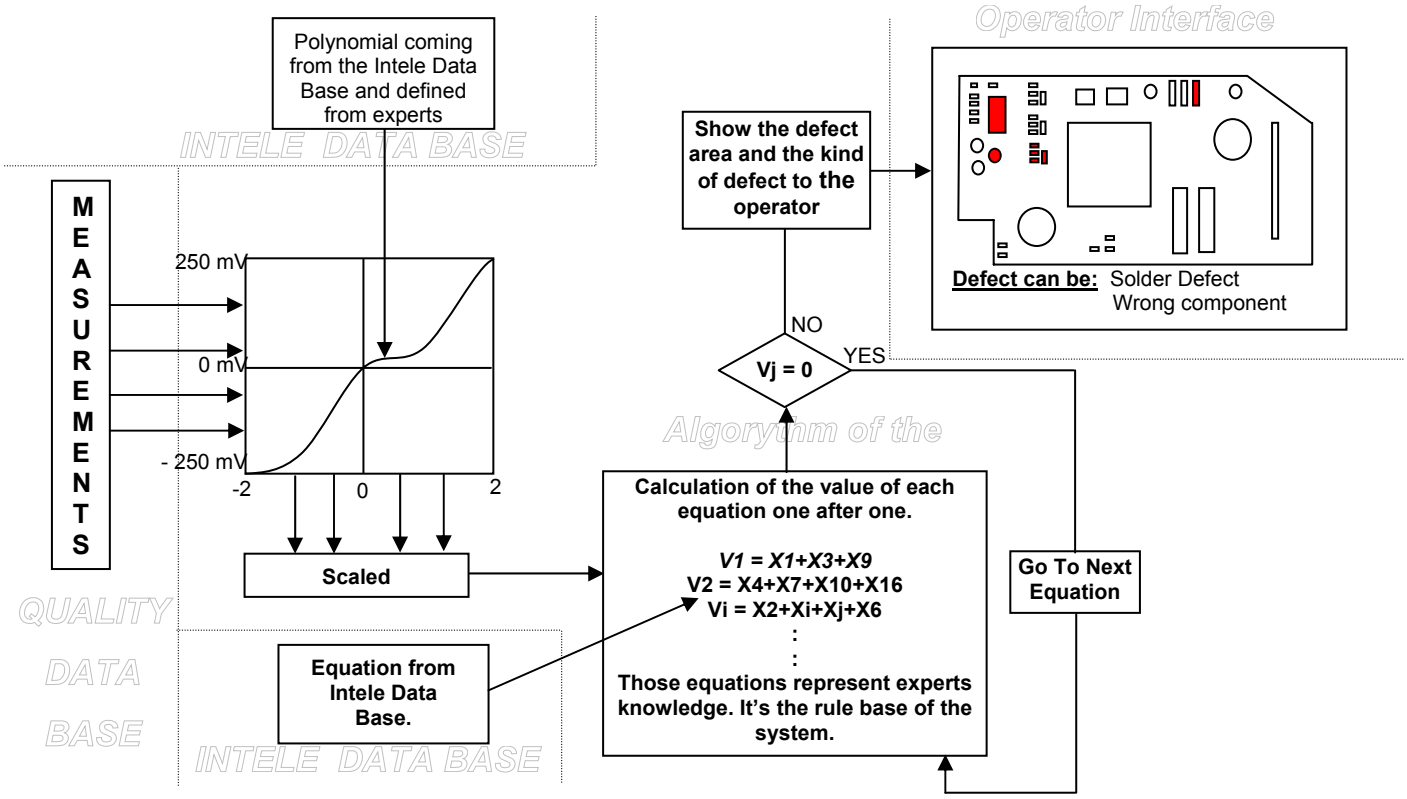

Figure 17: Software specification for operators.

This flow chart shows us how the system is working. It all starts with the measurements coming from the Quality database. Those measurements are then scaled according to the pattern relative to this product that is read from the Intele database. These scaled measurements become the linguistic variables for our equations. A simple test on the value of those equations allows us to identify defective areas on the PCB. Finally this information is sent back to the operators in a very visual way. This way, operators don't

need to care about complicated parameters. The easier the system is to understand, the higher will be the level of confidence of operators when they will finally have to use it.

5.3.2 PCB designers software

If the operators software was the mouth and the brain, our system is still blind at this stage. It needs eyes to collect information. The PCB designers software will have to fulfil this purpose by helping designers to put their knowledge in the database.

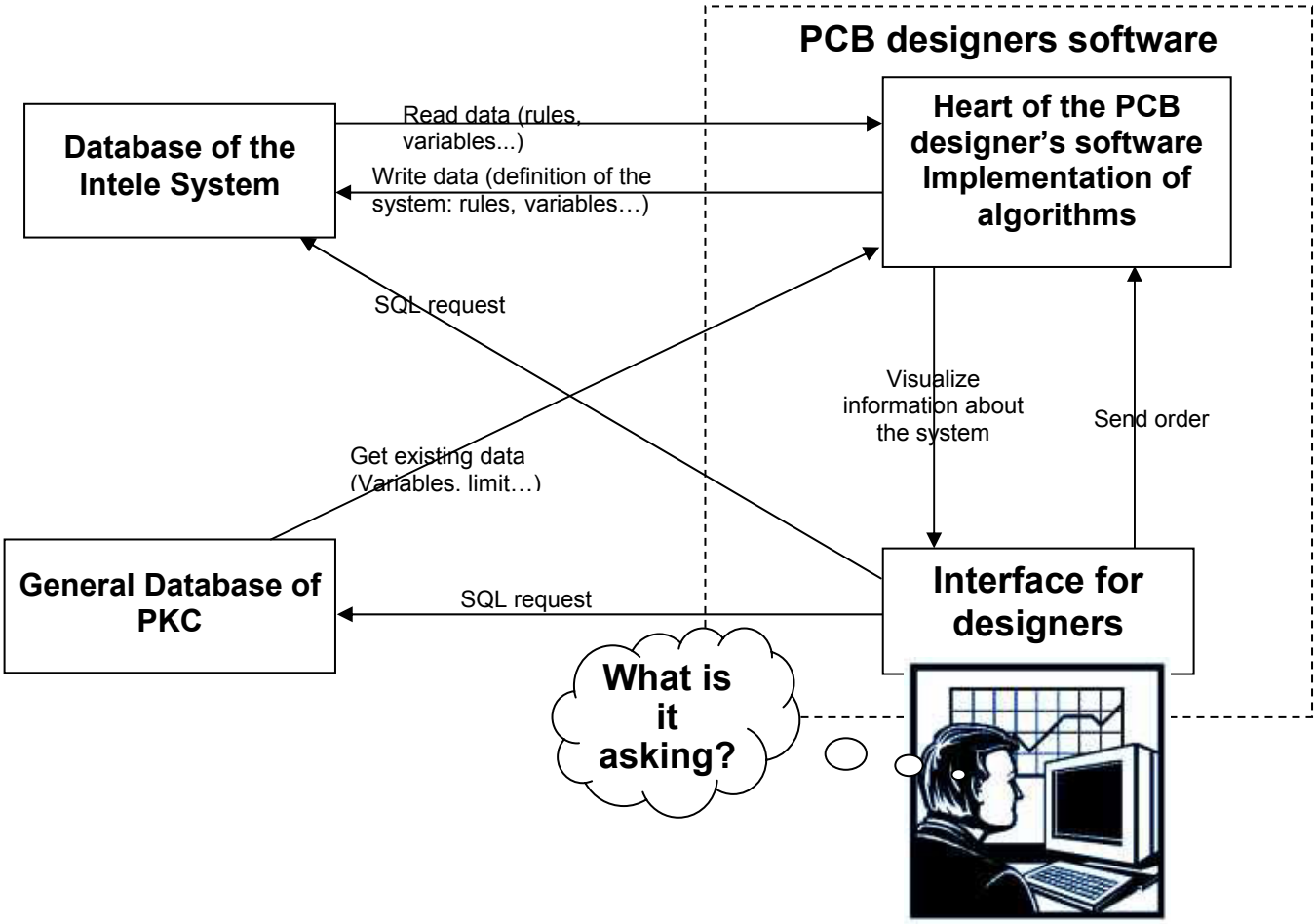

Figure 18: Software architecture for PCB designers.

This software (Figure 18) is mainly a communication tool between PCB designers and databases. The software is designed in such a way that a maximum information will be get and put to the database automatically. Nevertheless, information relative to defects or grouping of components in functional areas is the kind of information only available through designers. If the software is simple to use, maybe the designers will be able to focus on what they are doing instead on how they are doing it.

6. TOOLS USED FOR THE SOFTWARE DEVELOPMENT

Dynamic HTML and Visual Basic

DHTML and Visual Basic are two different tools. However they can be combined in an easy way to obtain interactive and powerfull software in a local intranet or even on internet. It is possible to combine the DHTML way of building a system with a simple architecture allowing us to navigate easily between entities, and the possibilities offered by Visual Basic in interface creation and algorithm implementation. The program can also be used by several users at the same time. Figure 19 for example is the page used for output definitions.

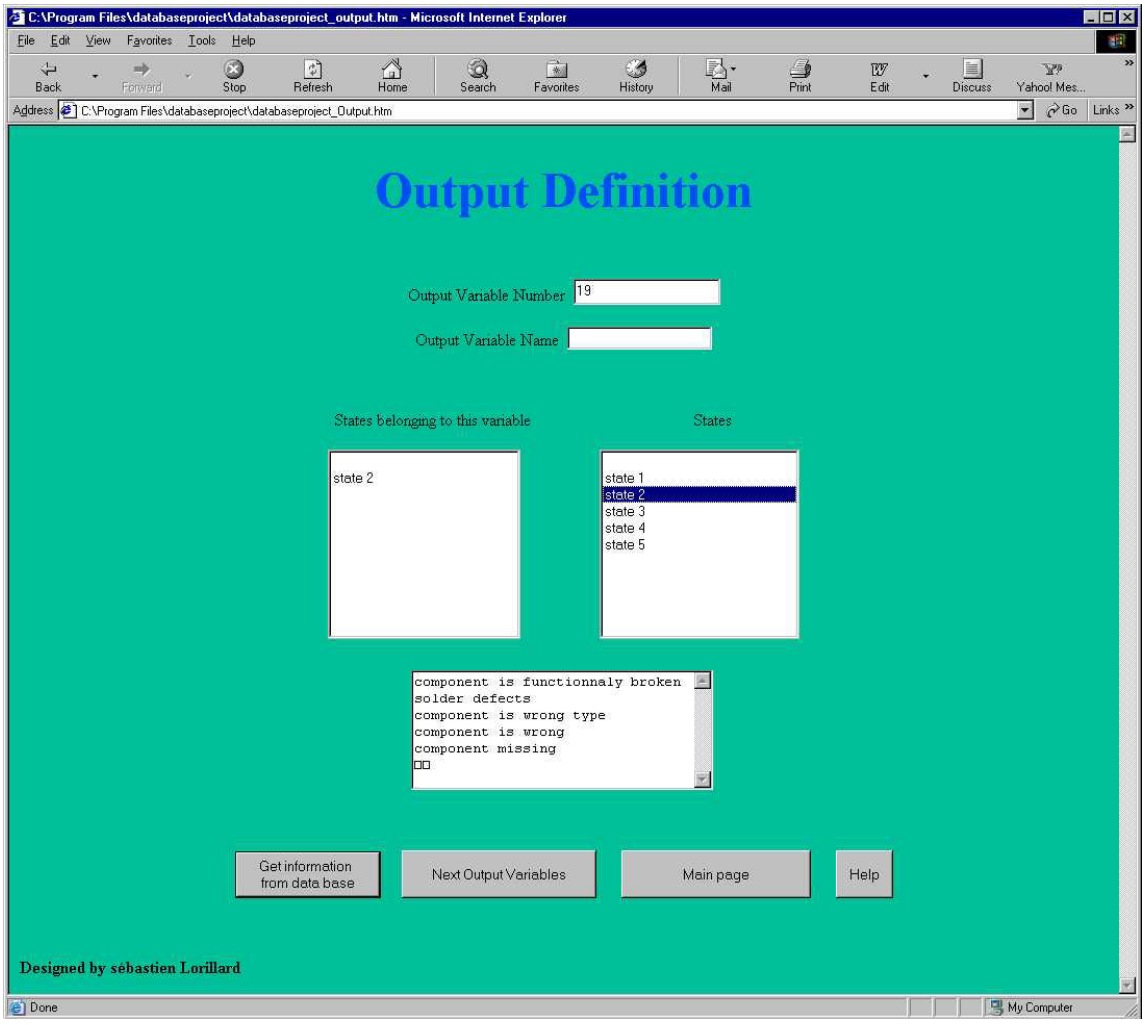

Figure 19: DHTML software.

With DHTML, we can build the software in a friendly and well known environement. Who doesn't know nowadays how a web browser is working? The resulting tool is intuitive and easy to use, meaning that it should fit with production constraints (flexibility, cost, delay).

ORACLE

PKC's production database is built on ORACLE. Besides being stable and reliable, it is easy to access with Visual basic.

7. A SYSTEM ORGANISED AROUND 3 ELEMENTS

As described earlier the system has three main parts. Two software systems, one for data acquisition for designers and one as an output interface for operators. The third part is a database representing the memory of the system.

7.1 Intele Database

This database has been built based on E/R models. First a draft version of the database has been made using Microsoft Access. This way, the entire database could be tested on a local computer, making modifications easier. Only once those tests were conclusive the database was build online with Oracle.

Oracle is a powerful tool but its lack of graphical interface makes it difficult to have a clear view of the database. It is therefore even more important to have a good E/R model for not getting lost and keep in mind the links between tables. For example, if we want to know the characteristics of the 3 components (C12, U4, D1) which are used in the product VBXXXX, the SQL request will look like this:

```
SELECT DISTINCT component.name, component.value, component.unit,
component.reference
FROM component, product, inputvariable, outputvariable
WHERE product.productnumber=inputvariable.productnumber AND
inputvariable.OVnumber=outputvariable.ovnumber AND
outputvariable.ovnumber=component.ovnumber AND product.name='VBXXXX' AND
component.name='U4'
```

The result of the request is the following screen (Figure 20).

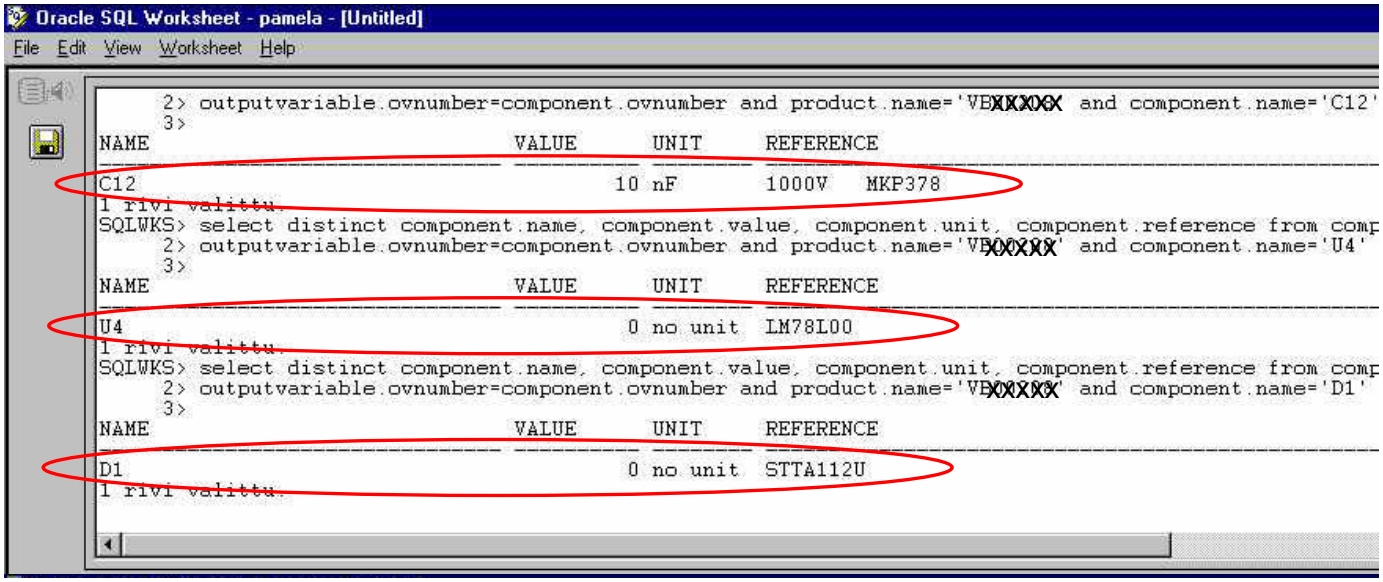

Figure 20: Result of an SQL request.

7.2 Designer knowledge acquisition software

7.2.1 General facts

The knowledge acquisition software is build around a main panel (Figure 21), where all the information that had to be defined for a product can be found. This information can also be loaded from the database and checked by the user.

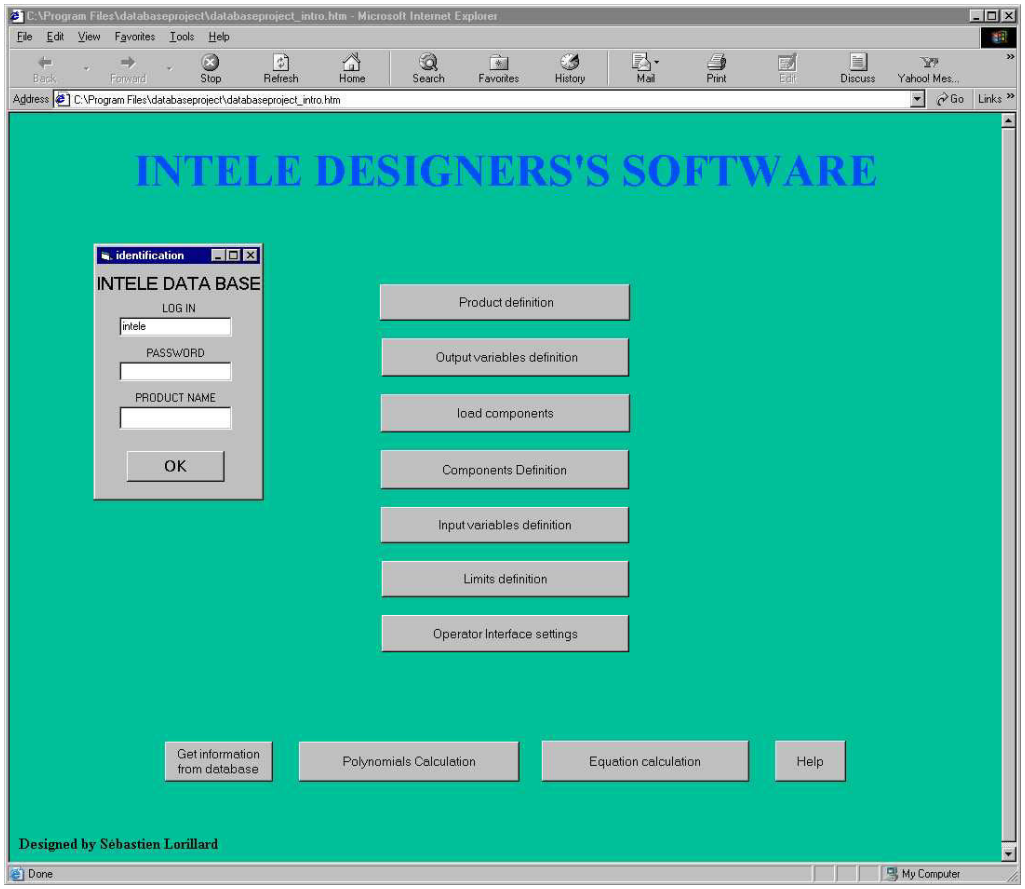

Figure 21: Main panel.

Features of the main panel

Product definition: It gives the product number, allowing us to retrieve all the useful information about that product.

Output variables: These variables represent functional area on the PCB. This part depends on the designer's knowledge of the product. Designers have to use their own functional understanding of the PCB to define areas. A good definition will have an

impact on the quality of the system's response. The better this definition is made, the better and more accurate will be the result.

Loading components: It loads the position of the components using files from the CAD tools or customers.

Components definition: After having defined the output variables, designers have to specify what component belongs to which area.

Input variables. These variables represent functional tests made on the PCB while it is on test stations. Besides that, designers also have to define the rules by linking input variables to output variables. It means that we assign the result of a test (state of an input variable) to a functional area (output variable).

Limits definition: The different states of each variable are defined here. Basically we are here dealing with tolerance limits for the tests and capability of the test station.

Operator's interface settings: Setting of some graphical parameters like scale and offset for the operator interface.

Calculation of the polynomes and the equations: This part is done automatically and is therefore transparent to the designer. Polynomes for example are calculated using test limits and measurements, whereas the equations are based on the rules linking input to output variables.

7.2.2 Example of component definition

Figure 22 shows an example of component definition. First part is to get the component names and their position from the Components loading panel. This is done by simply specifying the localisation of the file and selecting top or bottom according to the side of the PCB it refers to.

The component definition can now be done in the DHTML page. The definitions of the functional areas are made by double-clicking on the components to add or remove from an area. All those definition pages are built in a similar way, with four buttons in the bottom of the page. One is for the management module (section 7.2.3). The second one saves the information to the database. The third one is a link to the main page. And the last one opens a helpbox about the software. Here the use of DHTML pages proves itself very useful in the way the user can navigate between the different pages.

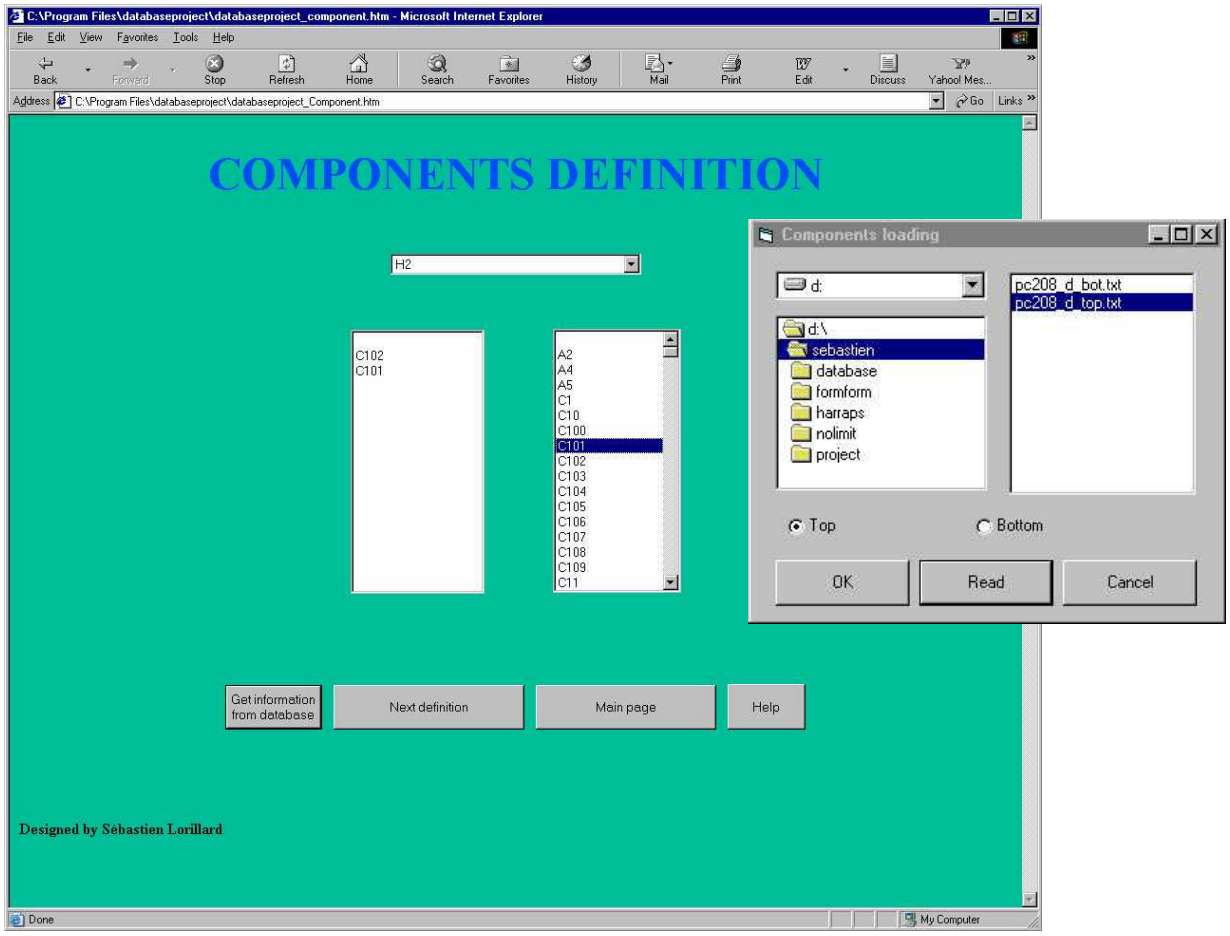

Figure 22: Component panel.

7.2.3 The database management module

The management module (Figure 23) has been added as a proofing tool. Through this module, the user is able to access all sort of information about all the products stored in the Intele database. All panels are designed in a similar way and it allows the user for example to get information from the current product or even older products and to delete wrong component definitions. Since some products also belong often to a same family, it is easy to copy the definition from one product to another one.

Figure 23 shows an example for the components panel. In this case, the selected product is VB00208 and the components panel wil give the position, side, and linked area for any component on this product. If the designer notices a mistake, he can delete the related component.

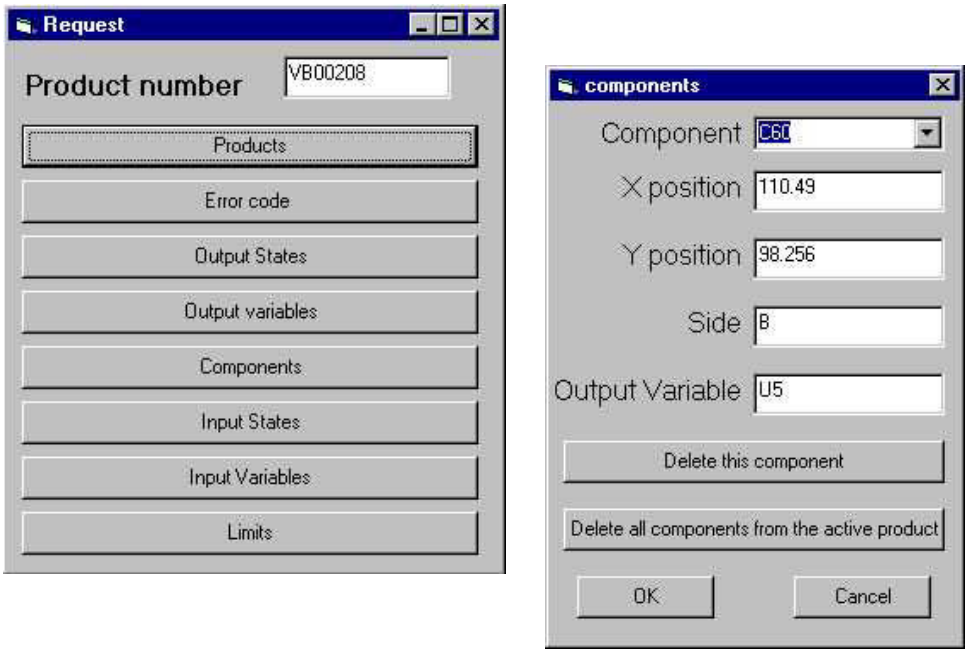

Figure 23: Management module.

7.3 Operator software

7.3.1 User interface

The main criterion for this software was to be easy to use for the operator. This goal has been achieved by reducing to its minimum the amount of information operators have to type in. Figure 24 shows the main panel and as you can see, the only information needed is the product number (selected from a list) and the test number of the PCB (given by test stations and written on the PCB). This information is enough to get a picture of the PCB where the possible defects are marked (Figure 25).

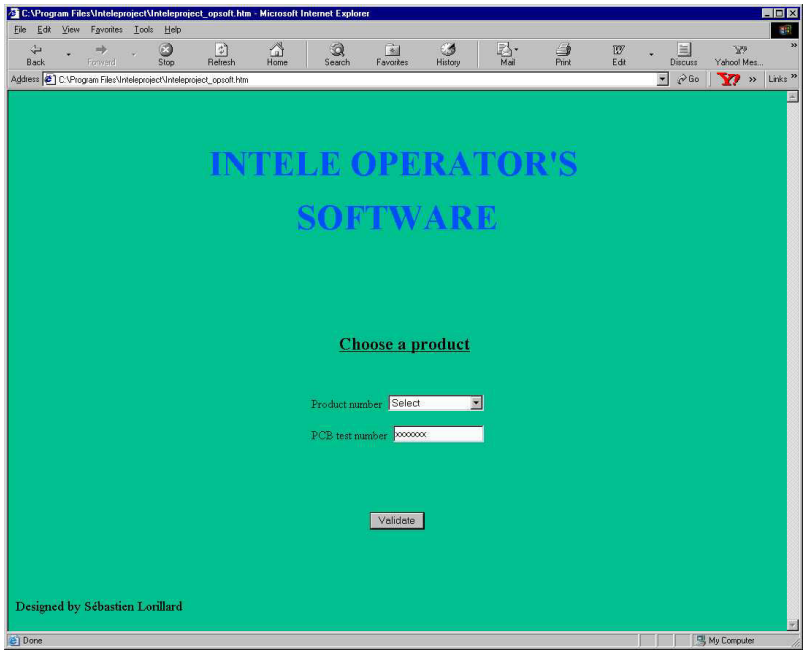

Figure 24: Software for the operators.

7.3.2 A visual result

Feedback about possible defects is given to the operator in a very simple and visual way that allows a fast comparison between the real PCB and the picture on the screen (Figure 25). A red cross marks possible defective components. It is also possible to get the name of those components by moving the mouse on one of them. Currently, because of a lack of defect information, it is not possible to point out a specific component. The crosses will therefore be put on all the components belonging to an area. However, by using historical data those crosses can change color reflecting better the higher probability for some components to be defectuous.

Last information given in this window is the name of the areas that can have defects and the kind of defect it can be (solder defect, wrong component…)

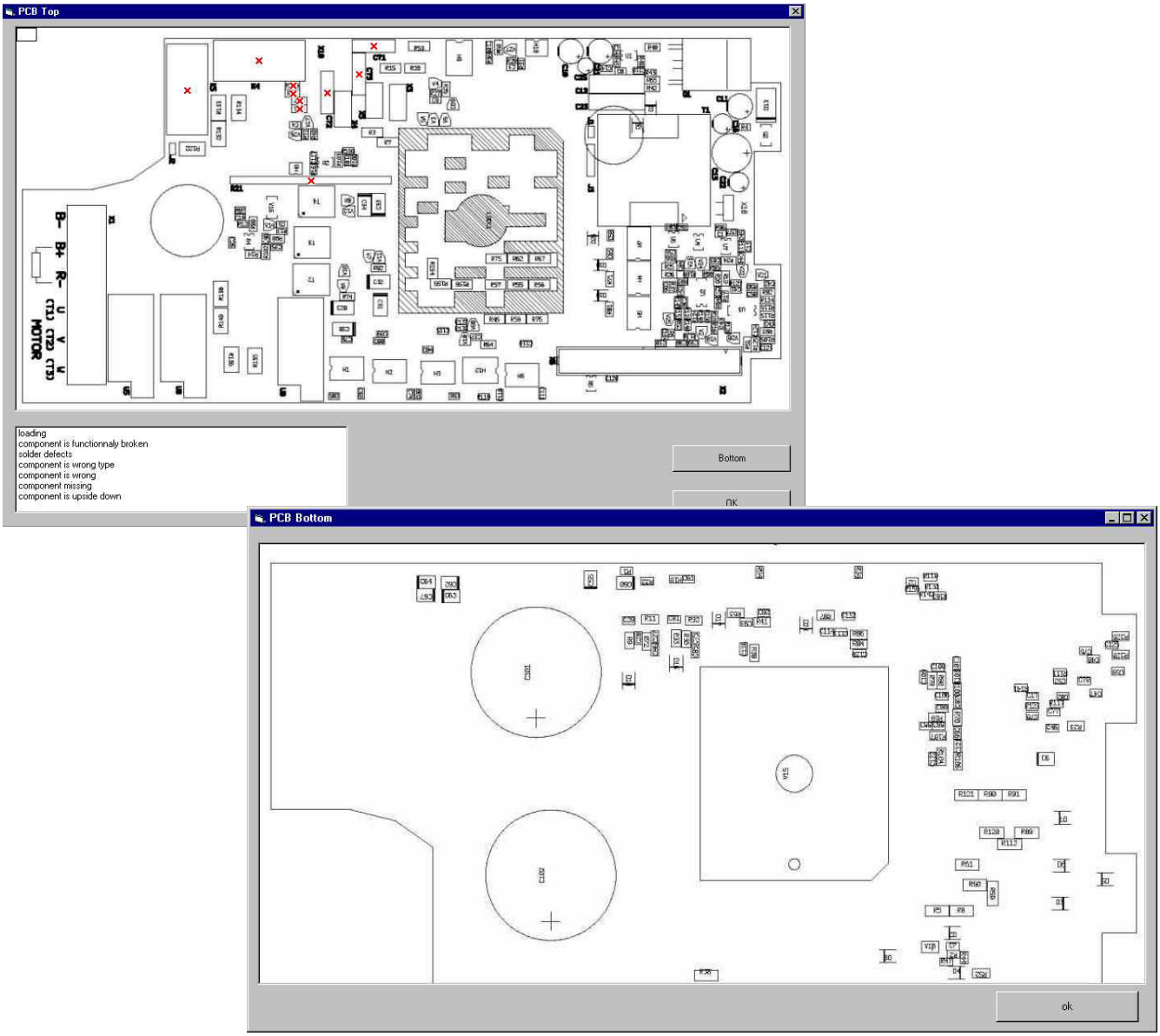

Figure 25: Result panel.

8. FIRST RESULTS

First tests of this decision-helping tool were made on a widely used product. This product had been produced already 7 months at the time of the tests and we had measurements for around 280 products. This means that polynomial regression and equation calculations were made with 280 measurements for each variable. Out of those 280 products there were 42 PCBs with errors. When tested with those 42 PCBs, the system gave the right answer for 95,24% of the cases.

In order to check the robustness of the system, we also made tests with good products. We obtain a rate of good answers of 95,39% of the PCBs for products without defects.

Last but not least, products in a prototype stage, with very few data available, have been tested. With data from 15 products (much less than one day of production) the rate of good answers was 90%. We can therefore assume that the ramp-up period for the system represents less than a day of production.

9. CONCLUSION

In the beginning, the project was very slow to start as the test an defect data did not meet the original expectations. In fact, it took nearly half a year to define guidelines and realistic targets. This required drastic changes in the development techniques: the original plan to use data-based system development had to be abandoned and replaced by an expertise-based approach. And even once this was done, the new targets only seemed realistic to us, but unfortunately not always to other parties involved in the project until they could finally see the first results. This shows the difficulty to conduct a project between the University and its need of scientific results, and a company whose interest is more oriented towards the final product. Having to deal with a “no matter what” approach on one side and a “no matter how” approach on the other side is not a pleasant situation. But we hope that with this work we were somehow able to reconcile those two approaches and prove that once the first difficulties and a certain level of scepticism are overcome it is eventually possible to do a work that will be appreciated by both sides. The scientific work was reasonable only on the basis of expert knowledge which was on the other hand available in an appropriate scale only after the first results.

This project has given us the opportunity to implement intelligent methods as part of a defect localization problem. The level of the problem that had to be solved was actually much lower than the real potential of those methods. Nevertheless, with only few applications nowadays of linguistic equations, it was interesting to see how it would be possible to adapt them to a problem with few degrees of freedom, in some ways even almost deterministic. With some minor changes, they proved to be a reliable tool with a major advantage, they are very simple to implement. It was possible to design a tool that doesn’t need any specific knowledge for the people who are using it because all the theoretical part is transparent to them.

According to the first results, the Decision-Helping Tool operates very well. The tool can be configured to new products, even during the ramp-up. On the other hand, the underlying methodology provides techniques for tuning the tool parameters when amount of testing data increases.

REFERENCES

1. **Balakrishnan, A., Semmelbauer, T., 1999**, "*Circuit diagnosis support system for electronics assembly operations*". Decision support systems 25, 1999, pp. 251-269

2. **Gebus, S., 2000**, "*Process Control Tool for a Production Line at Nokia*". Report A No 13, December 2000. University of Oulu, Control Engineering Laboratory, 27p. ISBN 951-42-5870-3

3. **Hyötyniemi, H., 2001**, "*Multivariate Regression – Techniques and Tools"*. Report 125, July 2001, Helsinki University of Technology, Control Engineering Laboratory. 227p. ISBN 951-22-5587-1

4. **Iserman, R., 1984**, "*Process Fault Detection based on modelling and estimation methods – A Survey*". Automatica, 20, pp. 387-404

5. **Iserman, R., 1993,** "*On the Integration of Fault Detection Methods via Unified Fuzzy Symptom Representation"*. In Proceedings of the First European Congress on Fuzzy and Intelligent Technologies –EUFIT'93, Aachen, September 7- 10, 1993}} (H.-J. Zimmermann, Ed.), Vol.1, pp. 400—407. Augustinus, Aachen.

6. **Isokangas, A., Juuso, E., 2000**, "*Fuzzy modeling with Linguistic Equations*". Report A No 11, February 2000. University of Oulu, Control Engineering Laboratory, 33 p. ISBN 951-42-5546-1

7. **Juuso, E. K., 1994**, "*Fault Diagnosis based on Linguistic Equation Framework*". In: T. Ruokonen (Editor), Preprints of IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes SAFEPROCESS'94 (Espoo, June 13-16, 1994), Hakapaino, Helsinki, 1994, pp. 374-379

8. **Juuso, E. K., 1997**, "*Intelligent Methods in Diagnostical Process Analysis*". Proceedings of XIV IMEKO World Congress, New Measurements – Challenges and Visions, Tampere 1-6 June 1997 (Jouko Halttunen, ed.), volume VII, pp. 1-6

9. **Juuso, E. K., 1999**, "*Fuzzy Control in Process Industry: The Linguistic Equation Approach*". In: Fuzzy Algorithms for control, International Series in Intelligent Technologies, pp. 243-300. Kluwer, Boston

10. **Juuso, E. K., 2000**, "*Linguistic Equations for Data Analysis: FuzzEqu Toolbox*". Proceedings of TOOLMET 2000 Symposium, Oulu, April 13-14, 2000, pp. 212-226, Oulun Yliopistopaino

11. **Juuso, E. K., Ahola, T., Oinonen, K., 1998**. "*Case-Based Reasoning with Linguistic Equations for Web Break Sensitivity Indicator*". Proceedings of TOOLMET '98 - Tool Environments and Development Methods for Intelligent Systems, Oulu, 1998, pp. 97-106.

12. **Komulainen, K., Heikkinen, M., Frantti, T., Juuso, E., Leiviskä, K., 1997**. "*Utilization of failure information in functional testing*". In L. Yliniemi and E. Juuso, editors, Proceedings of TOOLMET'97 - Tool Environments and Development Methods for Intelligent Systems, Oulu, April 17-18, 1997}, pages 109--115, Oulu, 1997. Oulun yliopistopaino.

13. **McKeon, A., Wakeling, A., 1989**, "*Fault Diagnosis in analogue circuits using AI techniques*". International Test Conference. Paper 5.2, pp. 118-123

14. **Stout, R.D.L., Van de Goor, A.J., and Wolff, R.E., 1998**, "*Automatic fault localization at chip level*". Microprocessors and Microsystems 22, pp. 13-22

15. **Tong, D.W., 1988**, "*Model-based reasoning for fault isolation*". Proceedings of the fourth Conference on AI for Applications, pp. 427-429

16. **Tong, D.W., Walther, E., Zalondek, K.C., 1989**, "*Diagnosing an analog feedback system using model-based reasoning*". Proceedings of the Annual AI Systems in Government Conference, pp. 290-295

17. **Ulieru, M., 1993**. "*From Boolean Probabilistic to Fuzzy Possibilistic Fault Tree Processing in Diagnostic Decision Making*". In Proceedings of EUFIT'93, Vol. 1, pp. 408--414.

18. **Wagner, L.C., 2001**, "*Failure analysis challenges*". Proceedings of the 2001 8th International Symposium on the Physical and Failure Analysis of Integrated Circuits, IPFA, pp. 36-41

Web:

Diagram Layout Conventions (SADT)
http://wwwis.cs.utwente.nl:8080/dmrg/MEE97/misop001/opg26.html

A structured approach to enterprise modeling and analysis (idef methods)
http://www.idef.com/

ACKNOWLEDGEMENT

The study of the Process Control Tool was done within the Socrates exchange programme between *University of Oulu* and *Institut Francais de Mecanique Avancee (IFMA)* in connection to the INTELE project funded by TEKES, PKC Group, Filtronic and JOT Automation.

ISBN 951-42-6731-1
ISSN 1238-9390
University of Oulu
Control Engineering Laboratory – Series A
Editor: Leena Yliniemi

1. **Yliniemi L, Alaimo L & Koskinen J,** Development and tuning of a fuzzy controller for a rotary dryer. December 1995. ISBN 951-42-4324-2.
2. **Leiviskä K**, Simulation in pulp and paper industry. February 1996. ISBN 951-42-4374-9.
3. **Yliniemi L, Lindfors J & Leiviskä K,** Transfer of hypermedia material through computer networks. May 1996.ISBN 951-42-4394-3.
4. **Yliniemi L & Juuso E** (editors)**,** Proceedings of TOOLMET'96 – Tool environments and development methods for intelligent systems. May 1996. ISBN 951-42-4397-8.
5. **Lemmetti A, Leiviskä K & Sutinen R,** Kappa number prediction based on cooking liquor measurements. May 1998. ISBN 951-42-4964-X.
6. **Jaako J,** Aspects of process modelling. September 1998. ISBN 951-42-5035-4.
7. **Lemmetti A, Murtovaara S, Leiviskä K & Sutinen R,** Cooking variables affecting the craft pulp properties. June 1999. ISBN 951-42-5309-4.
8. **Donnini P A,** Linguistic equations and their hardware realisation in image analysis. June 1999. ISBN 951-42-5314-0.
9. **Murtovaara S, Juuso E, Sutinen R & Leiviskä K,** Modelling of pulp characteristics in kraft cooking. December 1999. ISBN 951-42-5480-5.
10. **Cammarata L & Yliniemi L,** Development of a self-tuning fuzzy logic controller (STFLC) for a rotary dryer. December 1999. ISBN 951-42-5493-7.
11. **Isokangas A & Juuso E,** Fuzzy modelling with linguistic equation methods. February 2000.33 p. ISBN 951-42-5546-1.
12. **Juuso E, Jokinen T, Ylikunnari J & Leiviskä K,** Quality forecasting tool for electronics manufacturing. March 2000. ISBN 951-42-5599-2.
13. **Gebus S,** Process Control Tool for a Production Line at Nokia. December 2000. 27 p. ISBN 951-42-5870-3.

16. **Koskinen J, Kortelainen J & Sutinen R**, Measurement of TMP properties based on NIR spectral analysis. February 2001. ISBN 951-42-5892-4.
17. **Pirrello L, Yliniemi L & Leiviskä K,** Development of a Fuzzy Logic Controller for a Rotary Dryer with Self-Tuning of Scaling Factor. 32 p. June 2001. ISBN 951-42-6424-X.
18. **Fratantonio D, Yliniemi L & Leiviskä K,** Fuzzy Modeling for a Rotary Dryer. 26 p. June 2001. ISBN 951-42-6433-9.
19. **Ruusunen M & Paavola M,** Quality Monitoring and Fault Detection in an Automated Manufacturing System - a Soft Computing Approach. 33 p. May 2002. ISBN 951-42-6726-5.

20. **Gebus S, Lorillard S & Juuso E**, Defect Localization on a PCB with Functional Testing. 44 p. May 2002. ISBN 951-42-6731-1