HUNGARIAN JOURNAL
OF INDUSTRIAL CHEMISTRY
VESZPRÉM
Vol. 38(2). pp. 149-153 (2010)

MODELS AND METHODS TO DETECT SIMILARITY OF MANUFACTURING MACHINES

O. HORNYÁK[1]✉, G. SÁFRÁNY[2]

[1]Department of Information Engineering at University of Miskolc, HUNGARY
✉E-mail: hornyak@ ait.iit.uni-miskolc.hu
[2]evoPRO Ltd., HUNGARY

This paper focuses on detecting similarity of Manufacturing machines. It discusses the application of the Group Technology (GT) concept in the case of producing controller software for manufacturing machines. An overview on the modular machines is given. The paper presents a new concept for the development of the controllers' software. The advantages of GT are presented. Two grouping approaches are detailed: the bottom-up approach and the top-down approach. A Genetic Algorithm Clustering is presented for the automatic grouping. The second part of the paper deals with the similairity of PLC code. Smith and Waterman local allignment search algorithm is used to detect similar code paths in PLC programs

Keywords: Group Technology, Genetic Algorithm Clustering, Code similarity, Smith Waterman algorithm

Introduction

The design, assembly, starting up, monitoring and maintenance of industrial manufacturing devices have become very complex tasks over the time. According to the predictions for the future the complexity will not decrease. Simultaneously, the demands for and expectations of the quality of the products and processes have been growing. It is a well known fact that the cost price is an important factor in today's competitive manufacturing.

Another important factor is the lead-in time spent on the aforementioned activities. Engineers need to revise the activities required for producing a manufacturing design to be able to meet today's needs and to take a step towards future manufacturing devices.

The manufacturing machine producers need to develop the code used by their controllers. In case of process control the Programmable Logic Controller (PLC) based solutions are very widespread as the controlling algorithms are stored in a program.

At the moment the engineering tools used for creating such controller codes do not utilize the opportunities of Group Technology (GT), i.e. the reorganization and grouping of the similar hardware and software modules, and creating developer databases. So the engineering tools offer only a static help in the software development process. Industrial experience shows the fact, that highly qualified engineers are required for working confidently with such tools.

Modern software engineering tools have the capability to generate code automatically, see for example the recent releases of Developer Studio of Microsoft or Eclipse. Some Graphical User Interface (GUI) design tools or some database design tools can also generate code snippets or scripts.

Software engineering companies tend to have coding standards to enforce similar coding style among their staff.

Descriptions of algorithms are widespread through the internet; the implementations of these can be very similar.

The code copied from external sources can be either authorised or plagiarised.

Overview on modular machines

The design of manufacturing machines – both in the hardware and software aspects – is undergoing a sensible change. The solutions which offer customizations required by the merchant at the designing phase receive the attention of engineers.

- The use of modular building techniques – including mechanical parts, standardized communication, standard software modules, and generation of software modules – have been increased.
- It is expedient to use automated tools to develop such a customer-specific controller code which very often includes motion control modules as well.

Automated controller creation tools have the advantage of:
- increasing the precision of repeated tasks,
- reducing the dependency on engineers experience,
- improving the quality of the process and product documentation,
- reducing the preparation time of tendering,
- improving the flexibility of the preparation of the engineering work.

Automated generation of project variants

Besides the modular design of the manufacturing machines there is an increased need for implementing methods and tools by which the automated generation of the controller algorithms and project databases can be executed, especially without expert knowledge of the system. It is not necessary for the involved engineers to have an overall knowledge about the controller details. However, a generic change request or a call for a tender may require prompt action to develop the required machines, including the controller's software. Those machines are made to order or engineered to order, which means a high level of custom needs. To give a high service level for such customers there should be methods and tools for non expert engineers, marketing and sales staff which enable them:

- to combine the requested product (i.e. the generation of the product descriptor database, the automated generation of the controller code, the feasibility checks and availability of the required stock);
- to generate some software modules based on the existing project code library;
- to use an easy to understand, non technical workflow description language;
- to support the developer engineers in executing their repeated tasks;
- to support open source development.

Group technology

Group Technology is a management theory based on the principle that similar things should be done similarly. Products requiring similar operations, having similar attributes or needing a common set of resources are grouped into product families [3]. A key factor is the coding, which is an activity to create an – usually – alphanumeric key that describes the product and its attributes. The code must be exact and unambiguous. The literature identifies three kinds of codes:

- monocode (hierarchical code), where each character is restricted by the previous character,
- polycode (attribute), where each character has its own piece of information,
- hybrid (mixed code).

Feasibility of Group Technology for manufacturing machine controllers

The development process of creating a new manufacturing machine is a complex task. It is expedient to use the experience from former machine development projects. It is not an acceptable approach which does require the rewriting of an existing functional block of the controller software. Engineers need to split complex projects into small modules which can operate alone. Complex systems need to be built from these bricks according to predefined rules.

Fig. 1 depicts a development office, where engineers develop modules, hardware descriptor tables, and units made from modules. *Fig. 2* refers to the process in which new manufacture machine controller software can be automatically generated.

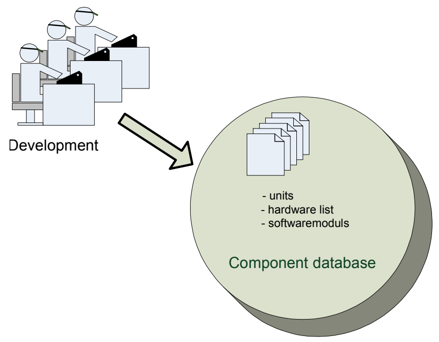


Figure 1: Development process of a new project

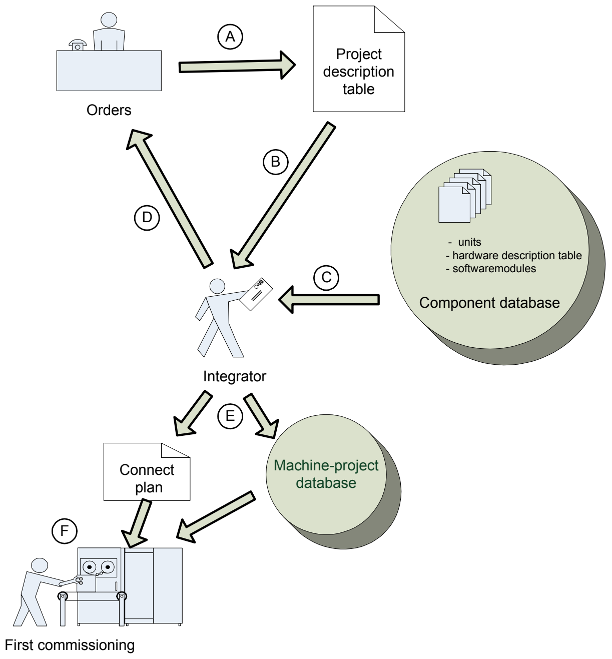


Figure 2: Simplified workflow of a project realization

Following the order A: the customer specific production machine description is presented. B: the technician doing the integration checks the components repository C for the required components in an automated way. Then a software tool generates the project database and the connected plan E for the machine controller. These passed to the installer staff F. After the integration there can be some feedback D.

Note, that the suggested development process implements the modules only once. These form a module repository so that the aforementioned advantages can be achieved.

When a new module is required to be developed the first step is to analyze the functionality. Experiments show that projects including similar modules have the same level of similarity. If the similar functions are implemented separately then it leads to redundant work and more opportunity of errors.

Engineers would need a tool which finds similar existing modules which implement the required functionality. The use of Group Technology can accomplish this to replace the human expertise.

As manufacturing machines have been created for many years now, Group Technology can be suitable for analyzing existing projects. Very likely a project analysis including module similarity check will exhibit a high level of redundancy.

In the following chapter the computer tools and mathematical models supporting these analyses will be detailed.

Automated clustering methods

Automated clustering methods should be computer algorithms with the capability of clustering the controller code of the machines. There can be two different approaches. The top-down approach uses the project description table and extracts the module information. Then it codes the functional components and tries to find similarity in the module/unit repository. This approach can be used during the tendering phase of a project, when the actual machine and its controller do not exist. The other approach is the bottom-up approach, where the controller code should be investigated. PLC coding can be made in various ways; some uses icons like Relay Ladder Diagram (RLD). Some codes have textual format like the Structured Text (ST) form. This is suitable for GT clustering.

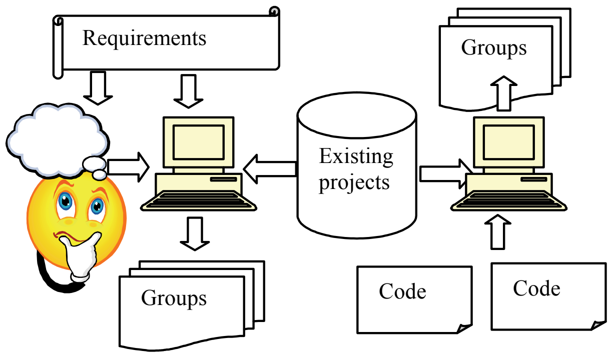


Figure 3: Top-down and bottom-up approaches

PLC code

The most widespread PLC programming languages are as follows:

CP – Contact Plan: a PLC programming language with the support of net-and relay switch plans.

FBD – Function Block Diagram: FBD is another graphical programming language. The main concept is the data flow that starts from inputs and passes in block(s) and generate the output.

LD – Ladder Diagram: Ladder Diagrams are specialized schematics commonly used to document industrial control logic systems. They are called “ladder” diagrams because they resemble a ladder, with two vertical rails (supply power) and as many "rungs" (horizontal lines) as there are control circuits to represent.

IL – Instruction List: Instruction List programming is defined as part of the IEC 61131 standard. It uses very simple instructions similar to the original mnemonic programming languages developed for PLCs.

ST – Structured Text: Structural Text is a high level PLC programming language such as Pascal.

SFC – Sequential Function Charts: Sequential Function Charts have long been established as a means of designing and implementing sequential control systems utilising programmable controllers. The Programming Standard IEC 61131-3 includes a graphical implementation of SFC’s in its suite of programming languages.

Mathematical formulization of the top-down problem

Let’s assume a binary polycode coding system. Each bit of the k bit long code represents a feature. 0 means that the controller does not implement the feature, 1 means that the feature is realized. Let’s investigate n projects. The problem can be represented by an n x k matrix M, see for example (1).

$$\begin{array}{c} 1\\2\\3\\4\\5\\6\\7\\8\\9\\10 \end{array} \begin{bmatrix} 0&0&0&0&0&1&0&1&0&0\\ 1&1&1&1&0&1&0&0&0&0\\ 0&1&1&1&1&1&0&0&0&0\\ 0&0&0&0&0&1&1&0&0&0\\ 0&0&0&0&0&0&0&1&1&0\\ 1&1&0&1&1&0&0&0&0&1\\ 0&0&0&0&0&1&1&0&0&0\\ 1&1&1&0&1&0&0&0&0&0\\ 0&0&0&0&0&1&1&1&0&0\\ 0&0&0&0&0&0&0&0&1&1 \end{bmatrix} \tag{1}$$

Let p_i represent the ith row. We can define a distance function by which we can evaluate how similar is one project to the other. For the sake of simplicity we assigned the decimal representation of the p_i vector. Let dec(p_i) denote this assignment.

Note, that this calculation is easy to execute, however it may give wrong result. For example

$p_1 = 0111111111$ is closer to
$p_2 = 1000000000$ than to
$p_3 = 0111111001$.

So clustering the distance function dec(p_i) will not give us a correct result, however, if we order the pi vectors according to their decimal representation then this can be a starting point for further clustering.

This method, however, will find it easily if two vectors are exactly the same: $p_i = p_j$ if dec(p_i)= dec(p_j).

Genetic algorithm clustering

The rows in matrix M need to be rearranged so that there will be block of 1s near in the p distance of the diagonal item. [4] details two measures of the goodness of the grouping, called Grouping Efficiency and Grouping Efficacy.

Grouping Efficiency is calculated as

$$\eta = q \cdot \eta_1 + (1 - q) \cdot \eta_2 \tag{2}$$

where η_1 is the proportion of 1s in the diagonal block, η_2 is the proportion of 0s in the off-diagonal block, q is a weight coefficient.

Grouping Efficacy is calculated as

$$\mu = \frac{N_1 - N_1^{out}}{N_1 + N_0^{in}} \tag{3}$$

where N_1 is the total count of 1s in the matrix, N_1^{out} is the count of 1s outside the diagonal blocks, N_0^{in} is the count of 0s inside the diagonal blocks.

A Genetic Algorithm Clustering (GAC) was implemented in the form of MATLAB code. Both the Grouping Efficiency and Grouping Efficacy were implemented as a fitness function.

The n number of rows and k number of columns had been split to y and x number of block $w_1, w_2, \ldots, w_y$ and $u_1, u_2, \ldots, u_x$ so that

$$\sum_{1}^{y} w_i = n \qquad and \qquad \sum_{1}^{x} u_j = k \tag{4}$$

If the GAC creates an instance which does not meet (4) then the fitness function will be 0. Otherwise the fitness function is calculated as described above.

The pseudo code of the GAC is as follows

```
Create the initial set of the population.
Evaluate each of them
While not to stop do
    Select two random elements
    Crossover random genes of those two
    Mutate the new specimen randomly
    Evaluate the new specimen
    The new population will be the top elite of the
    previous population plus the best ones of the new
    specimens
End while
```

After the block diagonalization process the (1) matrix should look as in (5).

$$\begin{array}{c} 8 \\ 6 \\ 3 \\ 2 \\ 4 \\ 1 \\ 9 \\ 7 \\ 5 \\ 10 \end{array} \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \tag{5}$$

You can see three groups identified by the algorithm here:

Group 1 consists of (8, 6, 3, 4),
Group 2 consists of (1, 9, 7),
Group 3 consists of (5, 10).

Computer algorithms for detection of similarity

Let's consider the bottom up approach, i.e. check the PLC code for similiarity. The Structured Text format discussed above is can undergo such an investigation because that is textual format of the code.

In the past, many computer algorithms were introduced for detecting software similarity, mainly for detecting plagiarism. [2] defines seven layers of code modification in a plagiarism spectrum. Some recommended using software metrics to detect similarity. [4] was among the pioneers: its metric used the following quantities:

n_1 = number of unique or distinct operators.
n_2 = number of unique or distinct operands.
N_1 = total usage of all the operators.
N_2 = total usage of all the operands.

Using these metrics we can calculate:

$$V = (N_1 + N_2)\log_2(n_1 + n_2) \tag{6}$$

$$E = (n_1 N_2 (N_1 + N_2)\log_2(n_1 + n_2))/2n_2) \tag{7}$$

where V refers to the volume of the program and E refers to the efforts to create the program.

You may find the logarithmic function odd at first sight. In this metric they found a correlation between the number of bugs in the program, programming or debugging time of the program and the complexity of the program.

Local alignment detection using Smith-Waterman algorithm

Smith and Waterman developed their algorithm to detect common molecular subsequences. They compared two molecular sequences:

$A=a_1\ a_2\ \ldots\ a_n$ and $B=b_1\ b_2\ \ldots\ b_m$.

The algorithm works as follows: set up a $H_{n+1\ m+1}$ matrix whose first row and column have the 0 index and are zeroed.

$H_{k0} = H_{0l} = 0 \text{ for } 0 \le k \le n$ and $0 \le l \le m$

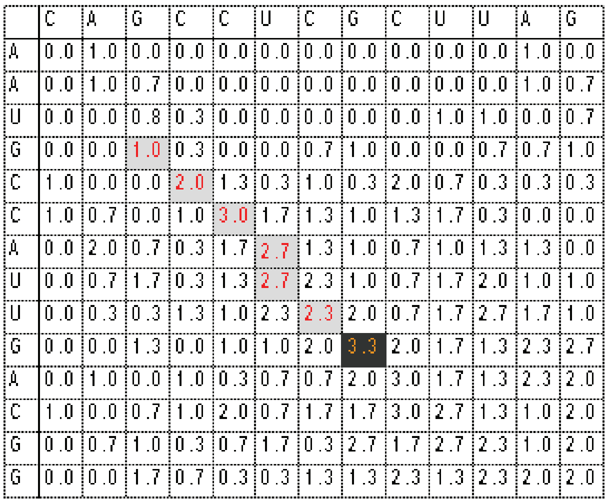
Then H_{ij} is the maximum similarity of two segments ending in a_i and b_j respectively. It is calculated as

$$H_{ij} = \max\left\{\begin{matrix} H_{i-1,j-1} + s(a_i, b_j), \\ \max_{k\ge 1}\left\{H_{i-jk} - W_k\right\}, \\ \max_{l\ge 1}\left\{H_{i,j-1} - W_l\right\} \end{matrix}\right\} \qquad (8),$$

where $s(a_i,b_j)$ is a score function for similarity, W_k is a weight (cost) function for a k-long deletion and W_l is a cost function for inserting l length of new sequence.

The highest score in the matrix indicates the maximum local alignment of the two sequences. Once the matrix elements are calculated the maximum element has to be found. That refers to the end of the maximum alignment. The traceback algorithm will find the way back. Find the next highest score.

– A diagonal jump implies there is an alignment (either a match or a mismatch).
– A top-down jump implies there is a deletion.
– A left-right jump implies there is an insertion.



	C	A	G	C	C	U	C	G	C	U	U	A	G
A	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
A	0.0	1.0	0.7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.7
U	0.0	0.0	0.8	0.3	0.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	0.7
G	0.0	0.0	1.0	0.3	0.0	0.0	0.7	1.0	0.0	0.0	0.7	0.7	1.0
C	1.0	0.0	0.0	2.0	1.3	0.3	1.0	0.3	2.0	0.7	0.3	0.3	0.3
C	1.0	0.7	0.0	1.0	3.0	1.7	1.3	1.0	1.3	1.7	0.3	0.0	0.0
A	0.0	2.0	0.7	0.3	1.7	2.7	1.3	1.0	0.7	1.0	1.3	1.3	0.0
U	0.0	0.7	1.7	0.3	1.3	2.7	2.3	1.0	0.7	1.7	2.0	1.0	1.0
U	0.0	0.3	0.3	1.3	1.0	2.3	2.3	2.0	0.7	1.7	2.7	1.7	1.0
G	0.0	0.0	1.3	0.0	1.0	1.0	2.0	3.3	2.0	1.7	1.3	2.3	2.7
A	0.0	1.0	0.0	1.0	0.3	0.7	0.7	2.0	3.0	1.7	1.3	2.3	2.0
C	1.0	0.0	0.7	1.0	2.0	0.7	1.7	1.7	3.0	2.7	1.3	1.0	2.0
G	0.0	0.7	1.0	0.3	0.7	1.7	0.3	2.7	1.7	2.7	2.3	1.0	2.0
G	0.0	0.0	1.7	0.7	0.3	0.3	1.3	1.3	2.3	1.3	2.3	2.0	2.0

Figure 4: Trace back from the maximum element

REFERENCES

1. O. Hornyak, G. Safrany: Group technology for automated generation of machine controller code. 5th International Symposium on Applied Computational Intelligence and Infromatics. (2009), Timisoara, Romania, 17–21.
2. J. A. W. Fahidi, S. K. Robinson: An empirical approach for detecting program similarity and plagiarism within a university programming environment, Compter Education, 11, 1987, 11–19.
3. A. Parker, J. O. Hamblen: Computer Algorithms form Plagiarism Detection, IEEE Transactions on Education, 32 (2), 1989, 94–99.
4. M. H. Halstead: Elements of software science, North Holland, New York, 1977.
5. T. F. Smith, M. S. Waterman: Identification of common molecular subsequences, Journal of Molecular Biology, 147, 1981, 195–197.