TAMPERE UNIVERSITY OF TECHNOLOGY

Department of Software Systems

Md Munir Hossain

Tool Support for Knowledge-based System Configuration in Factory Automation: A Connectionistic Grid Approach

Master of Science Thesis

Subject approved by Department Council

December, 2008

Supervisors: Adjunct Prof. Dr. Imed Hammouda (TUT)

Dr. Aleksandra Dvoryanchikova (TUT)

I. FOREWORD

This thesis work was done in 2008 in the Department of Production Engineering in Factory Automation Systems and Technologies (FAST) Lab.

I would like to thank Adjunct Prof. Dr. Imed Hammouda and Dr. Aleksandra Dvoryanchikova for their support during the thesis work. I would also like to thank Dr. Andrei Lobov for his support to my thesis work.

I would like to thank my family, all of my friends and well wishers here in Tampere for their help to accomplish this work.

Tampere, June 2011

Md Munir Hossain

Fysiikanpolku 2, B 28
33720 TAMPERE
Tel: +358 504 873817
md.hossain@tut.fi

II. ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY

Faculty of Computing and Electrical Engineering

Department of Software Systems

HOSSAIN, MD MUNIR: Tool Support for Knowledge-based System Configuration in Factory Automation: A Connectionistic Grid Approach.

Master of Science Thesis, 45 pages.

Examiner: Adjunct Prof. Dr. Imed Hammouda and Dr. Aleksandra Dvoryanchikova (TUT).

June, 2011

Keywords: System Reconfiguration, Connectionistic Grid, Ontology, Knowledge Representation, Factory Automation.

Traditionally manufacturing systems are controlled using Programmable Logic Controllers, which often require human intervention for system reprogramming on the arrival of a new product. In order to reduce all related costs associated with the human intervention, new systematic and automated system engineering approaches are needed. This thesis introduces a knowledge-based approach to the modeling and control of manufacturing systems aiming to capture the engineering knowledge. In order to demonstrate the applicability of the methodology, a software tool was implemented and applied to the case study, a pallet-based lifter used in electronics assembly, taken from the domain of factory automation. Early experiences show that the introduced approach can be used to capture knowledge pertaining to manufacturing equipments, processes and products. The approach could be considered as a potential solution for the implementation of reconfigurable control systems.

III. TABLE OF CONTENTS

IV. LISTS OF FIGURES AND TABLES

V. TERMS AND ABBREVIATIONS

User	A person who uses the tool
PLC	Programmable Logic Controllers
FOL	First Order Logic
NIST	National Institute of Standards and Technology
OWL-S	Web Ontology Language for Services
HTTP	Hyper Text Transfer Protocol
RRMS	Rapidly Reconfigurable Manufacturing System
XML	Extensible Markup Language
JDK	Java Development Kit
SOA	Service Oriented Architecture
DL	Description Logics
PSL	Process Specification Language
W3C	World Wide Web Consortium
WS	Web Services
CCG	Connectionistic Concept Grid
KR	Knowledge Representation
MVC	Model View Controller
UML	Unified Modeling Language
GUI	Graphical User Interface
SWSO	Semantic Web Services Ontology
DPWS	Device Profile for Web Services

1 INTRODUCTION

Industrialization is the base pillar of the developed economies as a mechanism for generating value. In Europe, manufacturing represents approximately 27.3% of GDP in 2010 according to IMF annual report. It is estimated that 75% of GDP and 70% of employment is related to manufacturing. Current market trends show that the demand is on the rise for highly customized products with ever shortening life cycle times [2]. And this trend is likely to excel in future. Customization has become a critical selling point for the products. The manufacturer that can provide more customization with less cost is more competent in the market. This market trend has a significant impact of the current manufacturing practices [2]. The manufacturing systems must be capable of dealing with a wide variety of products and must be able to cope with new types of products and processes with less human intervention.

1.1 Motivation

The manufacturing cost of a product depends a lot on the research and development cost and the system installment cost of the product. Therefore the production system is very important [3]. Moreover when a small customization is needed in the product, a lot of extra work needs to be done in the production system which adds cost to customized product at the end. That is why if a manufacturer thinks about customization of a product then he must think about the customization process in the production line in order to reduce the price of the product to keep his competent in the market. In this phase automation of customization comes in the picture because it can reduce the human intervention in the production system to facilitate customization to the product [4]. The more human intervention is needed for the customization the more the cost grows to the final product.

Automation is the use of control systems and information technologies to reduce the need for human work in the production of goods and services. In the scope of industrialization, automation is a step beyond mechanization. Whereas mechanization provided human operators with machinery to assist them with the muscular requirements of work, automation greatly decreases the need for human sensory and mental requirements as well [4]. Automation in this context means that the customization of the product is done without human engineer. At present required automation for customization can be achieved in mechatronics level using production system modules interacting with each other through their standard interfaces. But the software level system (re)configuration for these production system modules is not possible. To configure in software level in the modules still require human engineering knowledge [2]. The engineers rearrange the software modules and interface them according to the change done in the mechatronics level. The logic used in these levels is all known but it cannot be used to configure those software modules without human intelligence.

In order to autonomously reconfigure industrial devices and machines, engineering tasks currently performed by humans need to be automated [7]. Currently, engineers perform control development tasks based on their technical knowledge and on descriptions of manufacturing requirements, processes, machines and devices that are written in a combination of natural language and visual representation languages [2]. In order to automate engineering tasks, a machine-interpretable description of the engineering knowledge and the system specifications are needed in order to replace human-interpretable descriptions. To do this knowledge representation is required.

1.2 Objectives

Knowledge representation and reasoning is an area of artificial intelligence whose fundamental goal is to represent knowledge in a manner that facilitates inference (i.e. drawing conclusions) from knowledge [6]. It analyzes how to formally think - how to use a symbol system to represent a domain of discourse (that which can be talked about), along with functions that allow inference (formalized reasoning) about the objects. Generally speaking, some kind of logic is used both to supply formal semantics of how reasoning functions apply to symbols in the domain of discourse, as well as to supply operators such as quantifiers, modal operators, etc. that, along with an interpretation theory, give meaning to the sentences in the logic [8]. In the field of artificial intelligence there is a concurrent approach for knowledge modeling, which arose from Connectionistic theory and was first introduced in natural science for knowledge structure [6]. Connectionistic theory represented with symbol (shapes such as line, solid rectangle, overlapping shapes making intersection) can be a powerful approach to represent knowledge. In my research work that very approach is taken into use.

In this thesis work a software tool is developed which is used to represent knowledge using symbols and those symbols are following Connectionistic rules to reconfigure the software level in the factory automation system. The tool uses Connectionistic grid approach to capture and represent the knowledge which is provided through the graphical user interface it provides. User can use the tool and represent engineering knowledge through this tool and eventually can control software level configuration.

1.3 Structure

This thesis is structured as follows. Chapter two consists of the theory of system configuration. Chapter three describe software level system configuration in more details. Chapter four discuss about the theory of Connectionistic grid approach. The next chapter five discuss about the architecture, implementation and the user interface of the tool. Chapter six describes a case study. Chapter seven is the conclusion chapter describing advantage, limitation and the evaluation of the whole thesis work.

2 SYSTEM CONFIGURATION IN FACTORY AUTOMATION

The manufacturing processes that lead to the fabrication of a product typically span several organizations and create complex supply chains. The initial stages of manufacturing typically concern the assembly of pre-made components into specific product configurations. Fabricated components are often used in several product families and generations and therefore have relatively long production life-cycles [2]. However assembly processes need to be frequently changed for each kind of product. In view of the current market trends, it is in the assembly stages where mass customization and fast new product introduction can be achieved. The customization and fast new product introduction depends on the configuration of the assembly system. Conceptually the whole assembly system can be described as three level of configuration [5]. Those configuration levels are described in the following.

2.1 Mechanical Configuration

To understand mechanical configuration we need to understand what a mechanical system is. Mechanical system is consists of machines where a machine performs a specific task related to physical material, heat, pressure, force etc. Therefore a mechanical system can be seen as a system that takes raw material or semi-developed material and apply external force, heat, pressure etc. to develop further to a finished product [14].

A production system consists of one or more mechanical systems or tools where each doing a specific operation. Large mechanical system has been divided into modules of mechanical tools. Design standards have allowed the breakdown of a large monolithic mechanical system. No change can be introduced in work procedure in a large system if it is not modular. Moreover when the modular division is done then there must be a standard procedure of interaction among those modules to perform a seamless workflow. These procedures are defined by the specification. Modular machine tools have been on the market for a long time [15]. The international standards had been established to standardize the design and fabrication of modular units in the seventies [19]. In Figure 1 a schematic diagram of production modules are shown.

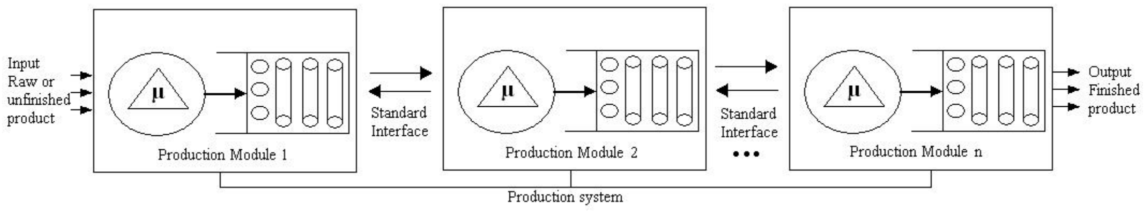


Figure 1: Production system modular division

In a manufacturing system to produce a specific product, a specific set of mechanical tool module is required with specific arrangement and a defined step of task that each module will do. Standard interface and standard communication protocol have facilitated the rearrangement of the production modules [4]. To develop a rapidly reconfigurable manufacturing system, production modules must be rapidly reconfigurable. Since there is standard protocol between the modules, it is not difficult to reconfigure those modules. The main challenge here is to configure those modules so that they can interact correctly according the latest configuration. That configuration is something that is done in the (Programmable Logic Controller) PLC of those modules. These PLCs are the control systems which drive the module to work correctly [19]. Configuration of the PLC is something that belongs to mechatronics configuration which is discussed next.

2.2 Mechatronics Configuration

According to French standard NF E 01-010 the definition of mechatronics is: “approach aiming at the synergistic integration of mechanics, electronics, control theory, and computer science within product design and manufacturing, in order to improve and/or optimize its functionality" [22].

Mechatronics is a multidisciplinary engineering system design that is a combination of Mechanical engineering, Electronic engineering, Computer engineering, Control engineering, and Systems Design engineering in order to design, and manufacture useful products. A part of mechatronics engineering is the study of programmable electronic device that can be used in a mechanical control system. PLC is a mechatronic device that is widely used to solve mechanical control system problem [19]. It is a digital computer used for automation of electromechanical processes, such as control of machinery on factory assembly lines, amusement rides, or lighting fixtures. A schematic diagram of mechatronics configuration is shown in Figure 2.

In the production system if the configuration in the mechanical tools module is changed then the PLC of that module is also required to reconfigure. This reconfiguration is done by reprogramming in the PLC. Early PLCs, up to the mid-1980s, were programmed using proprietary programming panels or special-purpose programming terminals, which often had dedicated function keys representing the various logical elements of PLC programs. Programs were stored on cassette tape cartridges. Facilities for printing and documentation were very minimal due to lack of memory capacity. The very oldest PLCs used non-volatile magnetic core memory.

More recently, PLCs are programmed using application software on personal computers. The computer is connected to the PLC through Ethernet, RS-232, RS-485 or RS-422 cabling [4]. The programming software allows entry and editing of the ladder-style logic. Generally the software provides functions for debugging and troubleshooting the PLC software, for example, by highlighting portions of the logic to show current status during operation or via simulation. The software will upload and download the PLC program, for backup and restoration purposes. In some models of programmable controller, the program is transferred from a personal computer to the PLC though a programming board which writes the program into a removable chip such as an EEPROM or EPROM [19].

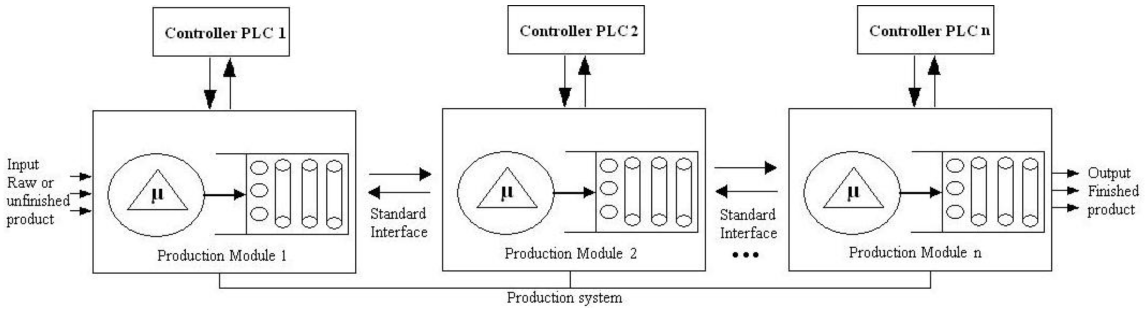


Figure 2: Production systems controlled by PLCs

2.3 Software Level Configuration

Software level configuration is the configuration among production modules in the production system chain. This level of control includes the underlying mechanical tool module PLC controlling and the overall production system performance, monitoring, ensuring the safety etc. The hardware level configuration is done through mechanical and mechatronics configuration using standard interfacing protocols. Software level configuration includes reprogramming of PLCs but not limited to only that [10]. A schematic diagram of software level configuration is shown in Figure 3.

The reconfiguration process for production customization depends on those mechanical module, PLC and software module reconfiguration. The task of software rewrite for the new operation is huge and requires heavily human involvement [29]. Therefore extras time and cost is required to do that in order to customize the product.

The intension in this thesis is to develop such a tool which can provide an interface that can be used by the operator of the system to configure the underlying software modules and the PLCs to eliminate human involvement of human engineer to reconfigure the system for product customization to work seamlessly with the changes done in the mechanical and mechatronics level.

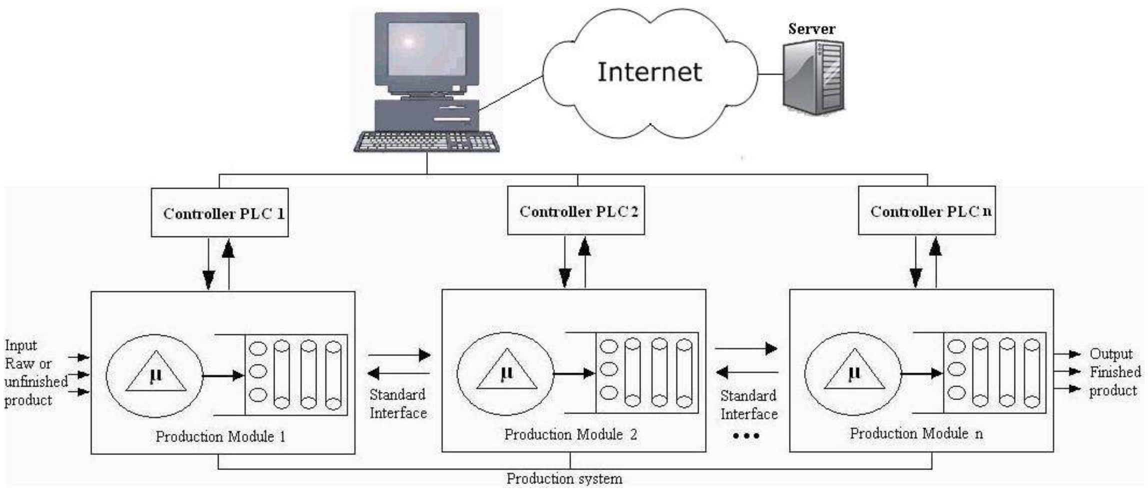


Figure 3: A complete Production system

The focus of this thesis is software level configuration in Reconfigurable Manufacturing System (RMS) in factory automation system. To understand completely what software level configuration means, we need to understand the re-configurability of the manufacturing systems and how this is done. We also need to understand the domain specific languages or knowledge that is required to reconfigure a manufacturing system.

As discussed in the first two chapters, RMS is a system which can be reconfigured to bring the customization to the product it manufactures. For that purpose the hardware mechanical modules and the PLC, the control system of those module need to be reconfigured [2]. Only these changes cannot ensure the manufacturing of the new product but some other higher level software control systems is changed accordingly. These software changes are a configuration management problem in computer engineering. The PLC is a special type of real time computer system that interacts with many input output channels simultaneously. The PLC has some domain specific language for programming and concept development. Those will be discussed in the following.

2.3.1 Configuration Management

In software engineering, software configuration management (SCM) is the task of tracking and controlling changes in the software. Software product is evolutionary in nature. It goes through a changing cycle from the time a software product is defined until it is no longer used and each change results in a different version of the product. Initiating, evaluating, and implementing the changes while maintaining product integrity is the purpose of configuration management.

In the context of factory automation, configuration management is very important for RMS. This is because RMS goes through changes all the time and the software of the system evolves into a new version every time.

2.3.2 Domain Specific Languages

In software development and domain engineering, a domain-specific language (DSL) is a programming language or specification language dedicated to a particular problem domain, a particular problem representation technique, and/or a particular solution technique.

Software development in the factory automation system is special software engineering fields where special types of programming languages are used and special type of modeling techniques taxonomy are used. IEC 61131-3 currently defines five programming languages for programmable control systems: FBD (Function block diagram), LD (Ladder diagram), ST (Structured text, similar to the Pascal programming language), IL (Instruction list, similar to assembly language) and SFC (Sequential function chart). These techniques emphasize logical organization of operations.

Currently all the necessary changes in the control system for product customization are done by human engineer. To automate the software configuration, human knowledge has to be modeled in a way that can be reused. This is called knowledge representation. A model of the assembly processes, performing devices, their interfaces and their effects is required for the product customization. This model must be able to present all objects of real world involved to manufacturing process and relations between these objects, facilitate reconfiguration at software level and must support decisions in choosing ways of possible solutions for current assembly tasks. One of the widely used methodologies of knowledge representation is Ontology. This method has been used in many computer systems for the formal representation of knowledge for many years.

2.3.3 Knowledge Representation Approaches

Representation can be defined as a “relationship between two domains, where the first is said to ‘stand for’ or take the place of the second” [1]. The former domain is often referred to as a model, and the discipline of representing a domain is referred to as modeling.

Visual modeling languages have been established as an important tool in control system development. In particular the Unified Modeling Language (UML) has gained acceptance as a tool to analyze and design control systems, helping developers in developing, reusing, extending and modifying software elements and systems [2]. However visual modeling languages are intended to be an aid to humans and not to automate engineering activities. In order to autonomously reconfigure mechanical and mechatronics devices, machine-interpretable description of the engineering knowledge and the system specification are needed. Knowledge representation techniques are discussed in the following.

Ontology

In computer science and information science, ontology is a formal representation of knowledge as a set of concepts within a domain, and the relationships between those concepts. It is used to reason about the entities within that domain, and may be used to describe the domain. In theory, ontology is a "formal, explicit specification of a shared conceptualization"[25]. Ontology provides a shared vocabulary, which can be used to model a domain - that is, the type of objects and/or concepts that exist, and their properties and relations [26].

Previously ontological knowledge bases were introduced with this aim to provide the semantic descriptions and facilitate the cognition [2]. The use of ontologies as a knowledge representation paradigm has recently gained significant attention and widespread adoption [5]. There are many tools which uses ontology to represent knowledge. Some of them are: Protégé, The Chimaera Ontology Environment, Open Cyc, OWL Ontology Validator, Ontolingua [Google directory].

Ontology represents knowledge as hierarchical representations of classes. The classes are represented as node and the interaction between two classes are represented by arc. However, this type of representation introduces a problem that is an object having same property may be named differently in different ontology representation. When two or more ontological representation is merged together then there is naming miss-match. In the manufacturing system, there are many vendors producing the hardware components; so, if there is a miss-match in the naming of the same hardware then the representation is not correct. Moreover, ontology represents a sequential model in human mind but the model required for the factory automation is parallel in nature.

Ontology has some drawbacks like: it appears to be limited in their capability to describe processes, which is a significant and fundamental part of the technical knowledge in manufacturing. Moreover it is not clear how to present the time flow of the processed in ontology. Next restriction lies in ways of association between concepts: there are mostly available the symmetrical ways of association. Moreover, the ontology is sensitive for errors. For instance, if one class is incorrectly spelled, it may lead to situation when all the following branch of descendants will be excluded out of the process. Another point of confusion is attributed to the integration of two ontologies in which the same objects have different name. It is difficult to extract expert's knowledge to present it in the ontological knowledge-base since expert's knowledge has a conflicting nature with many exceptions which are difficult to list in advance. These deficiencies motivate to develop such a tool presented here. Connectionistic approach of knowledge representation claim to overcome those limitations of ontology [6].

Connectionistic Grid

In the field of artificial intelligence there is a concurrent approach for knowledge modeling, which aroused from Connectionistic theory and was first introduced in natural science for knowledge structure [6]. The Connectionistic approach led to a number of impressive results in intelligent systems modeling [7] and it is mostly associated with artificial neural network. The Connectionistic networks are attractive because of their ability to simulate a complex reasoning. The main drawback of the approach application is less clear knowledge-representation schema.

Another issue in knowledge modeling is to find a proper level of formalization to allow the automated reasoning. As it was shown earlier by Lobov and Lastra that Edwards-Venn diagrams can be applied in reasoning for proving system correctness in the domain of factory automation. The diagrams are also applied here to support reasoning in Connectionistic concept grid. More discussion on Connectionistic concept is in the next chapter.

First-Order Logic

In mathematics, philosophy, linguistics, and computer science, First-order Logic (FOL), is used as a formal logical system. It is also familiar with other names, including: first-order predicate calculus, the lower predicate calculus, quantification theory, and predicate logic (a less precise term). First-order logic is distinguished from propositional logic by its use of quantifiers; each interpretation of first-order logic includes a domain of discourse over which the quantifiers range.

FOL is also known as First-order Predicate Calculus or Predicate Logic, enables representing the world by using propositions consisting of a predicate and one or more subjects. There propositions are believed to be unequivocally true, in contrast to modal logics (such as epistemic logic) that assigns degrees or modes of belief to the predicates. Reasoning in FOL is used to determine whether certain propositions, which are not explicitly given, also hold as true or not [2].

Expert systems, which gained popularity in manufacturing field during the 1980s and early 1990s, are based on FOL. Expert systems are generally equipped with a knowledge-base for a restricted domain. Even though expert systems appeared as a promising technology t replace experts in many domains, the industrial adoption of the technology has not been as high as originally expected. The main reason behind this is the high cost of the software development and the integration with factory equipment and especially high costs in terms of time and manpower required to develop a knowledge-base.

Web Ontology Language

Web Ontology Language (OWL) is an ontology language designed for use in the Semantic Web and is the language recommended by the W3C for this use. OWL has influences from quite a number of sources, but its main representational facilities are directly based on Description Logics. This basis confers upon OWL a logical framework, including both syntax and model-theoretic semantics. OWL also inherits from Description Logics a concern for practical reasoning and effective, readily available reasoners, for example the Description Logic reasoners Pellet and HermiT both of which have been extended to handle all of OWL [28].

OWL is also completely integrated into the Semantic Web and uses other W3C recommendations. The official transfer syntax for OWL ontologies is RDF/XML, a syntax for transferring RDF graphs. Names (of individuals, etc.) in the Semantic Web are IRIs, so names in OWL are IRIs. OWL uses XML Schema datatypes to define and type the data values that it uses. OWL ontologies are Web documents and are stored and retrieved just as other Web documents. OWL includes constructs for identifying ontologies and importing ontology into another. OWL ontologies can also be combined with rules using the new W3C Rule Interchange Format (RIF) standard [28].

2.4 Problem Description

With the aim to rapidly reconfigure the industrial devices of manufacturing systems, engineering tasks currently performed by human need to be automated. In order to do that human knowledge need to be represented in a way that is machine understandable. Therefore knowledge representation is the first step towards the goal. This research aims to apply Connectionistic grid approach which is derived from Connectionism theory to the domain of knowledge representation. A software tool has been developed to consolidate and apply that approach. Therefore the problem statement can be formulated as following:

How the Connectionistic grid approach can be applied in knowledge representation for the software level system configuration in factory automation systems.

2.5 Research Approach

A clear methodology is needed in order to perform a research. An extensive literature review is done related to the factory automation system. The current process flow and tool and techniques know-how are gathered to gain a solid background of the problem domain. These studies are based on the previous research work addressing similar type of problem from different perspective. Based on the study of the problem domain and the nature of the problem, probable correct solution theory is analyzed. The solution theory is analyzed from the perspective of its origin and the way and width of coverage for the probable solution. Based on the theory a suitable solution model is developed with certain assumption and limitation of scope. An implementation of the solution model is done and the result is evaluated against the goal.

An important note is to be made here is that, this research work has been done in the light of factory automation systems and the research work is presented here as a typical research work in software engineering department. The tool that is made out of this research is done from the scratch; so, there are clear limitations and these limitations come from both solution model establishment and the implementation of it. In this research there is discussion about the knowledge reasoning part but that is not taken into consideration in the implementation. This is simply because knowledge model has to be defined and implemented at first and then the reasoning can find its progress on that model. Since this is the beginning of the knowledge model establishment so the reasoning is kept aside from this thesis.

3 A CONNECTIONISTIC GRID APPROACH TO KNOWLEDGE REPRESENTATION

Intelligence of human being is one of the most complex and mysterious phenomena in the nature. Intelligence has a very close relationship with knowledge. We make decisions based on what we know about the world. Intelligence is something that makes the best use of the knowledge of the current world and decides the best possible action to take. So, we see that knowledge is the prerequisite to take the correct or in other word intelligent decision. Knowledge is an abstract term to us. We can say that knowledge is something that makes one agent to calculate the possibility of a proposition to be true.

Representation is also an abstract term to describe as knowledge. According to Brachman and Levesque [2], representation is a relationship between two domains. Where the first is meant to “stand for” or take the place of the second. Usually the first domain, the representation is more concrete, immediate, or accessible in some way than the second. The first domain contains some symbols and the second domain contains the meaning for the corresponding symbols and the relationship between these two domains builds the representation. In other word it can be said that a symbols represents a set of propositions. Therefore, the knowledge representation can be termed as the field of study concerned with using formal symbols to represent a collection of propositions believed by some agent putative agent. But it’s not guaranteed that the symbols must represent all the propositions believed by the agent.

3.1 High Level Concepts

Knowledge representation (KR) and reasoning' is an area of artificial intelligence whose fundamental goal is to represent knowledge in a manner that facilitates inferencing (i.e. drawing conclusions) from knowledge. It analyzes how to formally think - how to use a symbol system to represent a domain of discourse (that which can be talked about), along with functions that allow inference (formalized reasoning) about the objects [28]. Generally speaking, some kind of logic is used both to supply formal semantics of how reasoning functions apply to symbols in the domain of discourse, as well as to how to supply operators such as quantifiers, modal operators, etc. that, along with an interpretation theory, give meaning to the sentences in the logic.

3.2 Use of Knowledge Representation

Connectionistic approach may be a good technique to represent knowledge in factory automation process since it appears that, this approach can solve some of the limitations those ontology has. Connectionistic approach has not been used in factory automation system so far. Therefore, it is not well established how this approach can be applied best to this automation process. Connectionistic approach, which is derived from natural science, is a kind of parallel and distributed way of processing [18]. Since this is a very new application of this approach in this field, it is all unknown that how the knowledge structure can best match the problem domain of factory automation. Therefore, a software tool using Connectionistic approach is a possible way to represent human engineering knowledge and to configure the manufacturing components by providing some high level computer interface. This thesis presents such a tool using Connectionistic approach of knowledge representation. This very solution of fitting Connectionistic approach to factory automation for some research application is the main objective of my research work and eventually this thesis.

3.2.1 Set of Knowledge

The whole manufacturing process can be described in terms of three domains of knowledge [2]:

- Equipment,
- Process and
- Product

The concepts of the domains Equipment and Product are represented by nouns of natural language and the concepts of the domain Process are represented by verbs of natural language. Intersecting to each other the concepts of all three domains form the concept grid for an automated system.

The intersections of concepts build the simple concept grids which let to represent and control the current events and states of the system. Figure 4 shows a simple grid that could represent and reason a trivial assembling operation of perforating a detail. The concept Drill (Equipment) intersects with the concept To Drill (Process), which intersects with the concept Detail (Product). SUBJECT – PREDICATE – OBJECT.

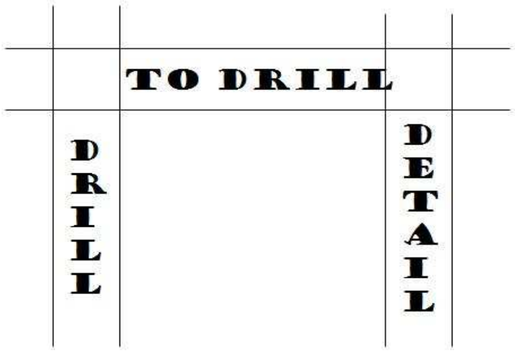


Figure 4: An example of a simple grid built of the concepts of three domains of knowledge [29]

To express an event or a state of the system the grid has to include minimum three concepts of at least two domains. One of the domain has to be the Process and another one or two have to belong to the Equipment or/and the Product (see for instance Figure 4 and 5). No grid is possible without a concept of Process; and no grid is possible with only concepts of Process. Thus, a simple grid can be built out of three concepts and two intersections between them, if one of concepts is from domain of Process and two other are from the domains of Equipment and/or Product.

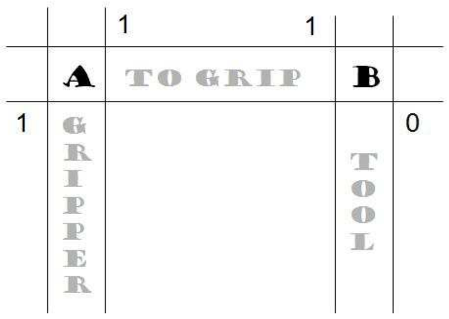


Figure 5: A grid of two domains with the bidirectional indexes of intersections [29]

It is important to express and reason asymmetrical relations between objects. For instance in the Figure 5 three concepts – Gripper, To Grip, and Tool are involved to the interaction. In common sense in real world a gripper could grip a tool, but vice versa situation is not any possible. With aim to manage asymmetrical relations the bidirectional indexes of intersections were introduced. In every moment of time every intersection has double figure number which shows the direction of Process. The concept is being a "subject" of the statement is indicated with "1". The verb in active form is indicated with "1". The "object" of action is indicated with "0". The verb in passive is indicated with"0". Thus every involved concept brings one figure to the bidirectional indicator of an intersection. For instance, to present and reason the operation "to grip the tool with the gripper" (Figure 5) bidirectional index of intersection A between Gripper and To Grip is "11" since both concepts are active and index of intersection B between To Grip and Tool is "10" since Tool is in the passive voice.

Four types of bidirectional indexes are possible in the grid:

11: Concept1 $\cup$ Concept2 $\neq \emptyset$

10 OR 01: Concept1 $\cap$ Concept2 $\neq \emptyset$

00: Concept1 $\cap$ Concept2 $= \emptyset$

The focus of this paper is assembly processes, which is a part of entire manufacturing. In this layer the functionality of the system can be expressed with three levels: Assembly Task, Assembly Process, and Assembly Operation [23]. In the CCG from Assembly Task to Assembly Operation every domain of manufacturing knowledge (Equipment, Process and Product) is represented with growing number of details. For instance, the concept Robot can be disclosed with concepts Joint and End-effector (Figure 6). All three concepts belong to the domain of Equipment. In this example Robot is a core of concept (CoC). Graphically the CoC is presented as a single strip in the grid, while the concept is always a complex pattern of activation. The dynamically changing pattern reflects and initiates changes in the system. Similar to directions of relations between the concepts from different domains (Figure 6), the direction of levels is shown with bidirectional indexes of intersection. For instance, in the Figure 5, both intersections have indexes "10", where "1" came from the CoC Robot and "0" came from its details.

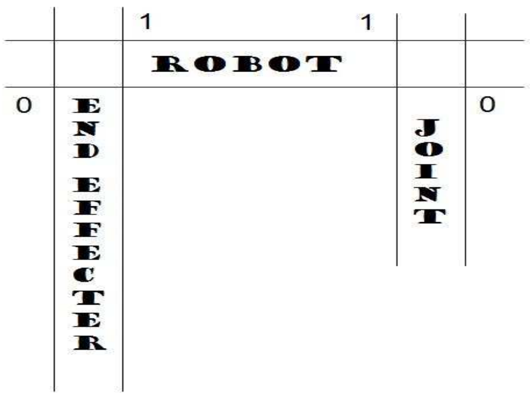


Figure 6: Representation of the domain in levels of manufacturing process [29]

3.2.2 Formal Notations

This subsection provides the basic definitions for the Connectionistic approach, which defines necessary structure to make it possible to implement the methodology. The Connectionistic approach deals with the concepts represented as a set of activated connections of the grid. The Connectionistic concept grid *CCG* can be formally represented as a tuple:

$$CCG = \langle CoC, A, P, AA, \alpha \rangle \tag{1.}$$

where:

- $CoC = \{c_1, c_2, \ldots, c_n\}$ is a finite set of cores of the concepts

- $A = \{a_1, a_2, \ldots, a_m\}$ is a finite set of activations defined for the *CCG* (representing actual concepts in the sense of symbolic approaches)
- $P \subseteq 2^{CoC}$ is a set of the activation patterns that are defined for the given *CCG*
- $AA \subseteq A$ is a set of currently enabled activations of the *CCG*
- $\alpha: A \rightarrow P$, is an activation function that maps an element of set *A* to the element of set *P*.

Set *A* can be dynamic in order to allow the learning process of the grid. For this the learning function must be specified for the grid. The learning concept grid *LCCG* can be defined as follows:

$$LCCG = \langle CCG, \lambda \rangle$$
(2.)

where:

- *CCG* is a concept grid as defined in (1.)
- λ: $2^{AA} \rightarrow a_{New}$, $a_{New} \notin A$, $A = A \cup \{a_{New}\}$; is a mapping of some active activations to the new activation that is added to the former activations set A to form a new activations set.

In the realization of the Connectionistic grid tool, the elements of the enabled activations set *AA* are mapped to the actions expressed in terms of I/Os of the controlled process and/or in terms of other activations.

The activations evolution law can be described as a function of currently enabled activations:

$$AA' = activate(AA)$$
(3.)

The activate function takes as an input a set of currently enabled activations *AA* and derives a new set *AA'*.

For the control of the physical process, the set of process inputs *I* and outputs *O* have to be specified. The definition of the *CCG* (1.) to address the control of the process can be extended. Control Connectionistic Concept Grid *CtrlG* is defined as a tuple:

$$CtrlG = \langle CCG, I, O, \iota, \omega \rangle$$
(4.)

where:

- *CCG* is a Connectionistic concept grid as defined in (1.)
- $I = \{i_1, i_2, \ldots, i_k\}$ is a set of control inputs
- $O = \{o_1, o_2, o_s\}$ is a set of control outputs
- $\iota: I \rightarrow AA$ is an input function that maps the control inputs to the activations.
- $\omega: AA \rightarrow O$ is the output function that maps the enabled activations of *CCG* to the control outputs.

The enabled activations of *CtrlG* are evolving through *activate* function (3.) and input/output functions defined in (4.) Execution model is described and shown in the next chapter.

4 TOOL SUPPORT: CROSSWORD

The tool “Crossword” is developed as a standalone application. It is developed using Java (J2SE) and swing UI package. Before going to the details of the tool architecture, execution model of the tool is presented in next subchapter. Architectural view and implementation details will follow subsequently.

4.1 Execution Model

In the previous chapter the Connectionistic grid theory is presented. From that theory equations (3.) and (4.), *activate* and ι and ω functions are implemented in execution engine in the following (Figure 7).

The factory automation system (FAS), which consists of different hardware and software components from different manufacturers using many standards and protocols, requires a logical hierarchy. In such layer architecture, each layer interacts with other through certain interface using certain protocol. Likewise to develop the tool a layer approach is adopted, where a chain of execution takes place in different layer. Figure 7 shows the execution chain model of three layers: model, execution engine and physical process.

Model: in the top layer, “Model” denotes the Connectionistic grid, which is developed in the tool by the user. This is a visual representation of a complex proposition which is applicable for the change in the underlying mechatronics level. The tool and the model are the main agents which take part in the top layer of the execution chain. The human user is interacting with the model through the tool’s visual interface. Construction of a visual model through the interface is a vital point to initiate the whole execution of the process.

Execution engine: The middle layer is a software layer which is called “Execution engine”. The main responsibility of this layer is to interface between the high level model and the underlying physical equipments. As the name suggests this layer is not at all responsible to create the visual interface. This layer has the instruction set; using these instructions it can instruct the machines to execute the activation of the high level model, ι, and it can also report the status of the physical equipments to the model by changing the activation of the model, ω.

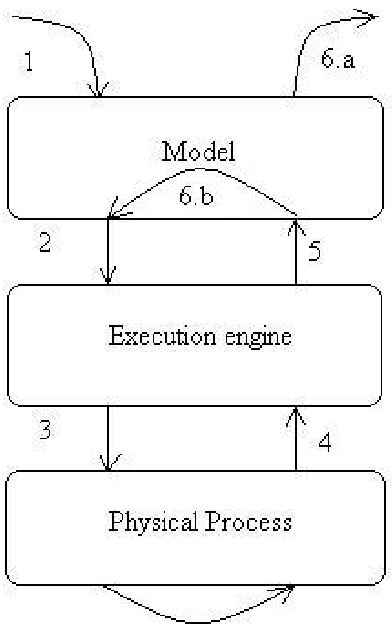


Figure 7: Model execution chain of FAS [29]

Physical process: this physical equipment layer is the bottom layer, where all the machines of the manufacturing system remain. This layer can take the command from the execution engine and can work accordingly and it can also report the status by performing some input-output I/O operations.

Different steps (step 1 to step 6) of the above figure are presented in brief as follows:

1. Activate the network (instruction given by the user).

2. The tool reports the activation of certain part of the model to the execution engine.

3. Execution engine converts the high level command to execute the activation for the physical equipment.

4. After executing the command the physical equipment will report the status by some I/O operations to the execution engine to transfer that status report to the high level model.

5. Execution engine will report the status of the physical equipments by changing the activations of different concepts in the model.

6 a. The user can observe the current status of the system and the system response in terms of activation of different concepts in the model.

6 b. The feedback from the hardware through Execution engine may activate new concept intersection, which will initiate the execution of the chain again.

4.2 Architectural View

The tool is developed following Model-View-Controller (MVC) design pattern. In this pattern "Model" is the logical representation of the data. "View" is responsible to display those data in the model and provide the possibility to edit and modify the data using proper user interface components. The controller holds the control of the system and propagates the events to the correct components.

Java swing framework is a widely used and matured widget toolkit for user interface development. This tool is an example of the use of this framework. Swing provides a native look and feel that emulated the look and feel of several platforms.

The tool implements equation 1 shown earlier and works in the "Model" layer of the execution model shown in Figure 7. In equation 1, *CoC* is defining a finite set of concepts. In the implementation of the tool these are actually some visual components shown in the visual interface. *A* is a finite set of activations defined for the *CCG*, which are again can be visible in the interface.

The tool implements visual interface for the user and also calculates the set of concept and intersection evolving dynamically using the specification (1). Here the implementation uses MVC design pattern, where the view part is responsible for the visual interface display and model holds the actual data those are required in equation 1. The controller is communicating with view, calculating model data and interfacing with the "Execution Engine" as shown in execution model. In Figure 8 the package diagram of the tool is presented.

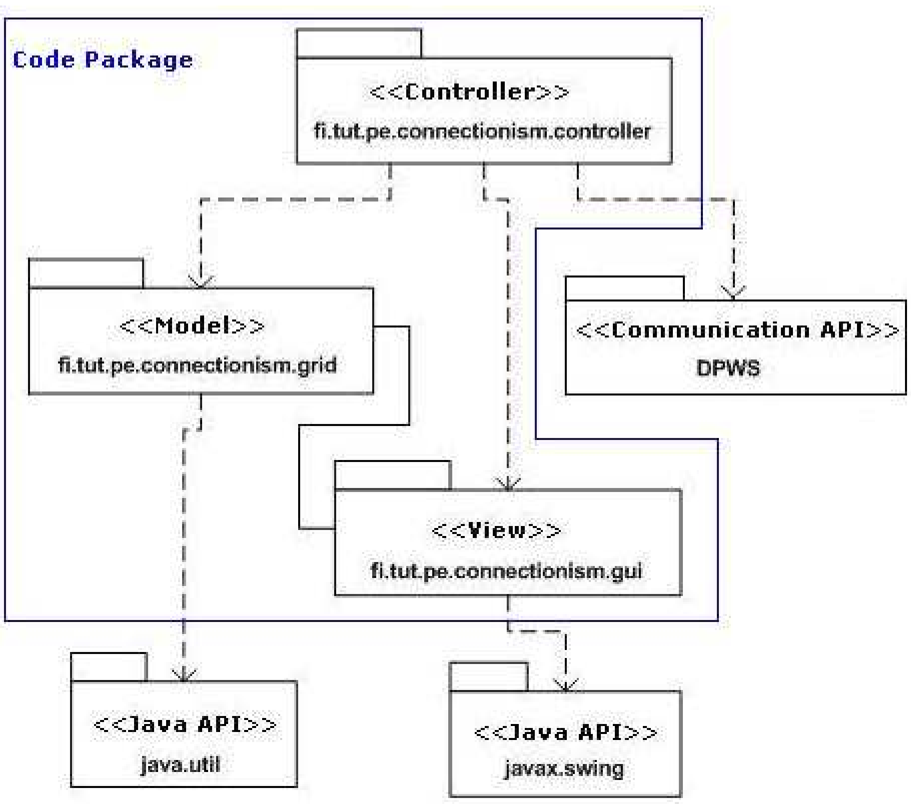


Figure 8: Package diagram of the tool

In the package diagram, *grid* package is holding the model and the *gui* is holding the view classes. There is a link between these two packages because some visual representation value is transferred into the model. The *controller* package uses all three packages. The *DPWS* (Device Profile for Web Services) package is a third party open source java implementation [21] SOA4D, DPWS implementation, https://forge.soa4d.org/, [*Accessed: November 2008*] for the communication with the physical process. For implementation of the tool, java standard edition is used along with the java swing package for drawing the concepts and intersections in visual interface. The view classes are derived from the classes of swing package. The models classes are composition of own classes and java util package classes as shown in the package diagram.

javax.swing and *java.util* packages are good enough for the purpose of the development. Though they do not provide many advanced features, they provide the basic necessary features to implement view package of the application.

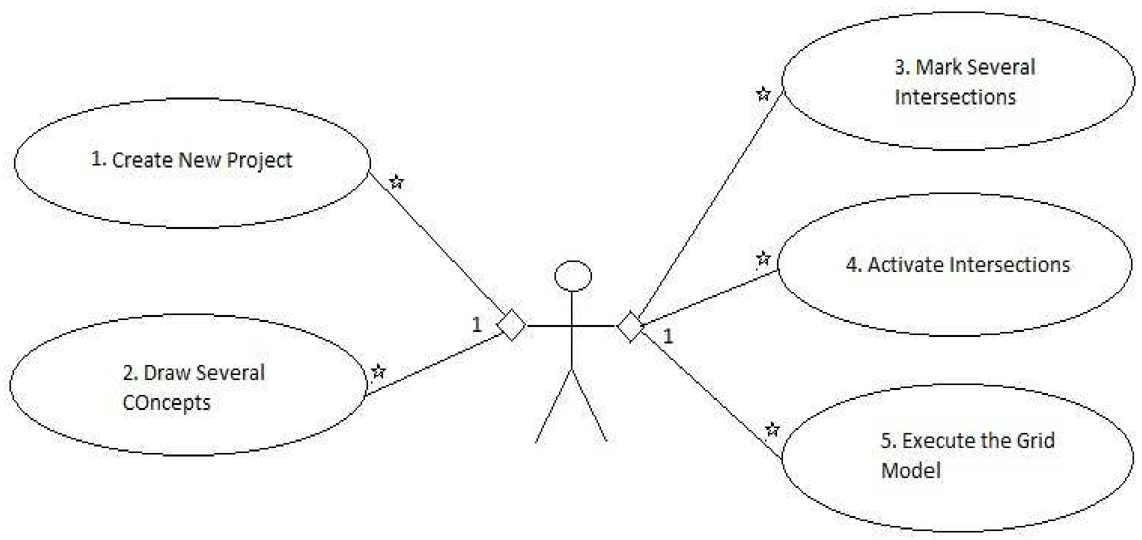

1. Create New Project
2. Draw Several COncepts
3. Mark Several Intersections
4. Activate Intersections
5. Execute the Grid Model
1
1
☆
☆
☆
☆
☆

Figure 9: Use case diagram of the tool

The interface of the tool is described in the case study chapter with short description of the interface components. The interaction between the user and the tool is described by a use case diagram shown in Figure 9. The diagram shows the major use cases for the tool. Every *CCG* is developed in a project, so, first task is to create a project. Then the concepts and intersections are drawn according to the need of the underlying hardware. After that the intersections are activated and the execution of the grid model starts.

The software tool to be developed is a single threaded desktop application. It is not the requirement to develop it as a desktop application but for simplicity it is done so. This tool is intended to interact with the PLCs of the manufacturing equipment. The tool is following the Model View Controller (MVC) design pattern having a Graphical User Interface (GUI) for the user. The user should be able to develop the *CCG* in a GUI environment. The tool's user interface is like that of file manipulation tool's interface, where the user can change the properties of a file for the file browser. In the case of the tool, the user should be able to change the properties of the concept from the visual interface of the tool. From this usability perspective of the tool, it can be modularized as: GUI module for the user interface, communication module for tool to machine's PLC communication, inference module for concept and intersection reasoning from the defined rules and the controller module for controlling all the module and work in a synchronized manner.

4.3 Implementation

This sub-chapter details about the implementation of the tool. It starts with the geometric theory and then how those theories are used in the tool. Geometric theories and drawing is at the heart of the implementation of this project. Some code snippets of the tool have been given at the end of the sub-chapter.

4.3.1 Geometric Theories

For this tool implementation the most important symbol is the line. Connectionistic grid consists of symbols. These symbols are nothing but polygon and the polygon is basically consists of lines. So the geometrical element line is the most important theory where for drawing. Equations of lines such as parallel line, perpendicular line and intersecting points of two lines and the dividing angle between the lines are very important. Those theories will be discussed in the following.

Straight Line

Here line means two end bounded straight line having equation of a line is $y = mx + c$ in the Cartesian plane; where m is the slope or gradient of the line, c is the y intercept of the line and x is an independent variable of the function $y = f(x)$. Figure 10 shows the lines in the Cartesian plane with red, blue and green color.

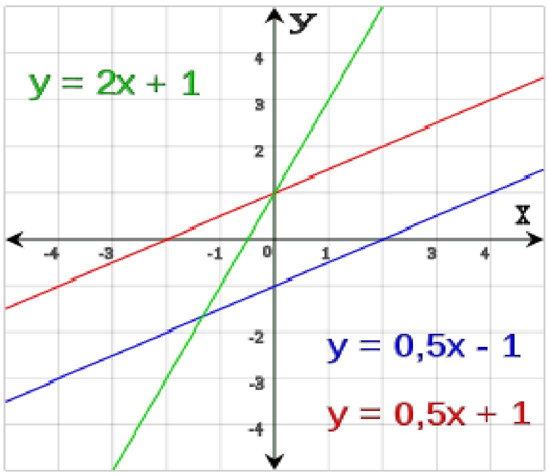


Figure 10: Straight lines in Cartesian plane

Parallel Lines

If two straight lines do not intersect each other even they are hypothetically enlarged infinitely at both ends then those two straight lines are called parallel to each other. In Cartesian plane, if a line PQ having the equation of $y_{PQ} = mx_{PQ} + c_{PQ}$, then a line parallel to another line, RS will have the equation of $y_{RS} = mx_{RS} + c_{RS}$. Here the slope or the gradient of both of the lines are same, which is m. Figure 11 shows two lines a, b and another line t intersecting both of them. Parallel lines generate exactly same angle with the intersecting line.

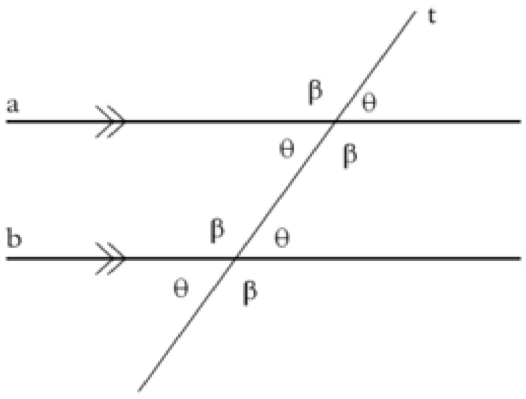


Figure 11: Parallel lines in Cartesian plane

Intersecting Point between Two Lines

If two straight lines intersect each other then they will have a common point which is the intersection point. Figure 12 shows the intersection point of two lines.

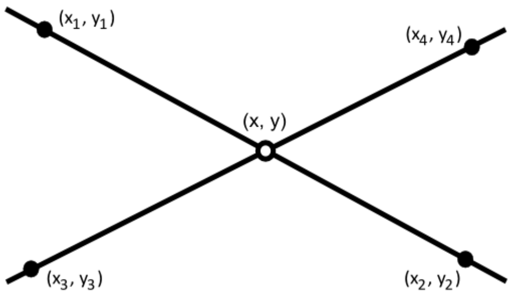


Figure 12: intersection point of two lines

The equation of the intersection point of two lines shown in Figure 12 is:

$$(Px, Py) = \left(\frac{(x_1y_2 - y_1x_2)(x_3 - x_4) - (x_1 - x_2)(x_3y_4 - y_3x_4)}{(x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4)}, \frac{(x_1y_2 - y_1x_2)(y_3 - y_4) - (y_1 - y_2)(x_3y_4 - y_3x_4)}{(x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4)}\right)$$

Angle Bisector

An angle bisector divides the angle into two angles with equal measures. An angle only has one bisector. Each point of an angle bisector is equidistant from the sides of the angle. The interior bisector of an angle is the line or line segment that divides it into two equal angles on the same side as the angle. The exterior bisector of an angle is the line or line segment that divides it into two equal angles on the opposite side as the angle. Figure 13 shows the angle bisection.

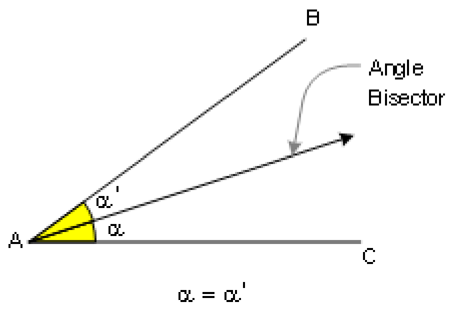


Figure 13: Angle bisector

4.3.2 Application of Geometric Theory in the Tool

In the Connectionistic grid there is symbol called concept, which are actually special type of polygon from a drawing perspective. The use of the tool creates the symbols by mouse clicking on the viewing panel and this polygon is drawn on the panel according to the mouse button pressed. For instance, if the user presses mouse button on three points *(A, B, C)* then the tool generates a polygon shown in Figure 14.

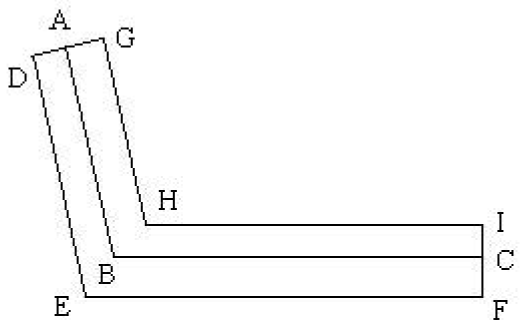


Figure 14: a sample concept drawing

This polygon is a shape drawn around the line connecting those mouse press points *A*, *B* and *C*. The polygon to be drawn is *ADEFCIHG*. To draw the polygon, all the corner points are to be determined. Here *DEF* and *GHI* is equidistant to the *ABC* line. Finding those corner points are purely geometrical calculations.

The principle that is followed to draw the polygon is to draw a line that is connecting those points of mouse click at first. Then two perpendicular lines at the both end of the line have to be drawn. In the Figure 14 those perpendicular lines are *GAD* and *ICF*. Those perpendiculars are to be drawn so that *DA* is equal to *AG* and *FC* is equal to *CI*. Then *DE* and *GH* parallel to *AB* and *FE* and *IH* parallel to *CB* are calculated. *DE* and *FE* are intersecting each other at point *E* and *GH* and *IH* are intersecting at *H*. In this way all the six points are found for three points mouse click. These six points are the corner points of the polygon. The line *ABC* is not drawn in the view but the polygon *DEFIHG* is drawn which is actually a much thicker line of the original *ABC* line. Later in this chapter small code snippet is given to show how those points are drawn in the actual tool implementation.

In a Cartesian coordinate system two straight lines *L* and *M* may be described by equations.

$$L : y = m_1 x + c_1$$

$$M : y = m_2 x + c_2$$

as long as neither is vertical. Then m_1 and m_2 are the slopes of the two lines. The lines *L* and *M* are perpendicular if and only if the product of their slopes is -1, or if $m_1 m_2 = -1$.

The perpendicular line equation and the intersection of two lines are at the heart of concept drawing. The algorithm of drawing concept is given below.

```
DrawConcept
    Input: A list of mouse points clicked.
    Output: Points of the polygon to draw a concept.

    for each pair of points
        find a straight line
      //e.g. for A, B, C points find AB, BC lines. These lines are also considered as main lines.

    for each line
        find two parallel lines keeping a constant distance 'd'

    for each pair of parallel lines
        find the intersection points

    for each end points // e.g. for A, B, C points A and C
        find transverse lines

    find the intersection points for both end transverse lines
    find the intersection points for the parallel lines

    put the intersection points in a vector in order

    return the points vector
```

The above algorithm can be understood with the help of the following Figure 15. If the user clicks in four points (*A, B, C* and *D*) then three main lines are constructed as *AB*, *BC* and *CD*. Then three pairs of parallel lines and two transverse lines at points *A* and *D* are calculated. Now, these parallel lines and the transverse lines intersect each other. These intersection points are calculated in order. Intersection points are calculated for lines which stay in the same side of the main lines. This is very important because this maintains the order of the points. Actually at the end when the concept is drawn in the drawing panel, it is drawn as a polygon. The intersection points are the real polygon points. In the following picture the concept (the polygon in geometrical perspective) is shown filled with green color.

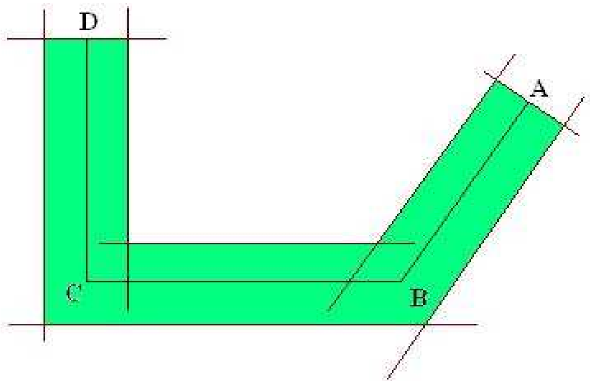


Figure 15: How the algorithm calculates the corner points of a Concept

In Connectionistic grid there is another symbol called intersection. When two concepts intersect each other, then the intersecting geometry is called the intersection. In Figure 16 *UVWX* is the intersection of two concepts. The intersection is calculated using the equation of finding intersecting points of two straight lines as shown in subchapter Intersecting Points between Two Lines. Calculating the intersecting points and finding the intersection geometry is not all. It is also important to know which concept is intersecting which concept. In Figure 16, Concept 1 is intersected by Concept 2, which is why the intersection color is the same of the color of Concept 2.

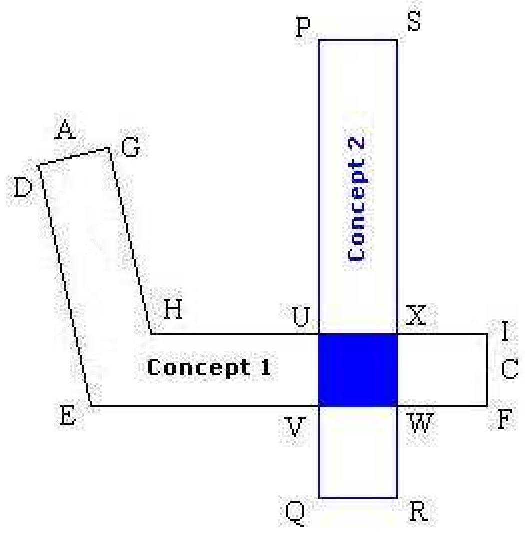


Figure 16: Intersection of two concepts

4.4 User Interface

The tool's interface is a standard WIMP (Window, Icon, Menu, and Pointer) interface with menu bar and tool bar. The tool bar is inactive in the beginning. The general use case of the tool is to make a new Concept Grid at first: to do that user goes to *File* menu and *New* submenu as shown in Figure 17.

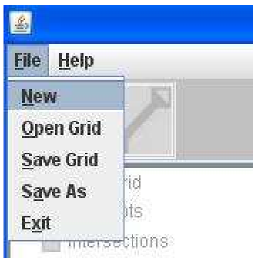


Figure 17: *File* > *New* to develop a Concept Grid

User is then prompted with an *Input Dialog* to give the name of the Grid as shown in Figure 18. After the Grid name is provided, the main user interface is active and ready for the user to draw the Concept Grid: the tool bar and the drawing panel are activated as shown in Figure 19.

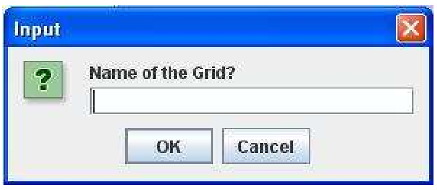


Figure 18: input grid name

The user interface of the tool is divided into five sections, shown in Figure 19. They are described below:

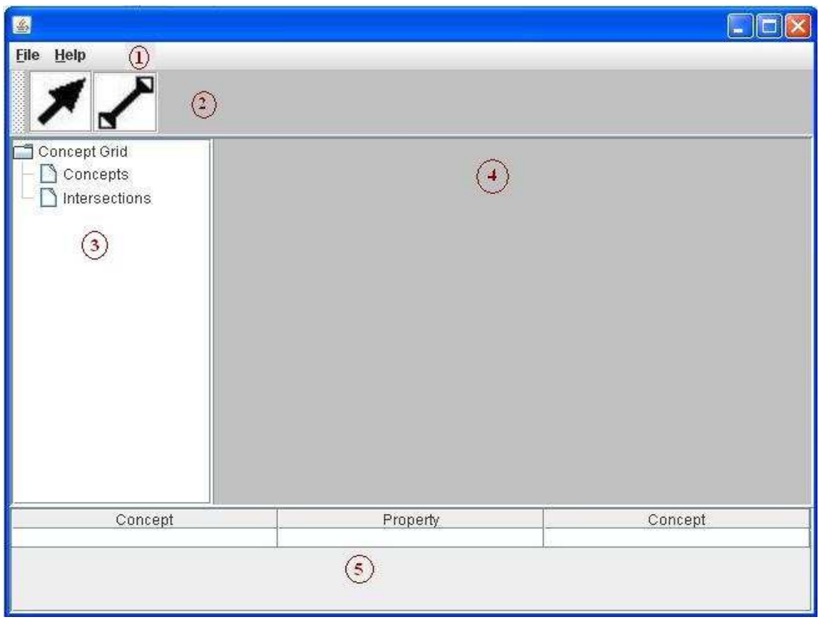


Figure 19: UI of the tool

Menu bar: in Figure 19, menu bar is marked as number 1, this contains several menus such as: create *New* concept grid, *Save* the grid, *Open* the previous saved concept grid or *Save as* different grid. *Help* for general help for the tool and copyright declarations.

Tool bar: in Figure 19, tool bar is marked as number 2, which contains two tool buttons. One is to draw Concepts and the other is to mark intersection of concepts as Concept Intersection. The arrow labeled tool is for drawing Concepts and the other one is for Concept Intersection.

Tree perspective: it shows every Concept and the Concept Intersection that belongs to a Concept Grid in a tree structure. It helps the user to select concept or intersection easily and perform several operations such as *Rename*, *Activate*, *Deactivate* and *Delete*.

Drawing Panel: this panel is the geometry where the user draws the Concepts or Concept Intersections. User at first presses the *Concept* button from the tool bar and then clicks in the panel in different points as the concept should look like. Then at the end he or she clicks the right mouse button to mark as the finishing of the drawing. Each mouse click point will be connected by a line then the concept will be drawn as mentioned the previous implementation subchapter.

Relationship Table: this table shows the inter-relationship of the Concepts and Concept Intersections. The example is shown in the case study chapter.

The tool stays at the top level of the execution chain. The user of the tool can develop the model using the tool. The model is defined as Connectionistic grid. The tool can validate the model. The validation means that the operator cannot define any model which violets some predefined rules. For example, if the rule is such that a pallet cannot be lifted up in the middle of loading a product on it; then if the user defines the model where the motor for lifting the pallet can be started in the middle of the pallet uploading – then the user violates the rule. In this case the tool will notify the user.

The user selects first activation to initiate the execution of the whole process (step 1).

In step 2, when the model is activated by the user, the activation is reported to the execution engine by the tool. There is a software interface between the tool and the execution engine. By this interface the tool can notify the activation of the model which is basically some instruction to be performed by the physical process.

In step 3 the execution engine transfers the activation change of the model by generating proper instruction which is understandable to the physical equipment. The machines will receive the command and execute them to report the status after the execution.

In step 4 the physical equipment generates the message to report the status after the execution of the previous command given by the execution engine. These are I/O operations understandable by the execution engine.

In step 5 the execution engine generates proper activation to report the current status of the physical process, which was reported by them in step 4. In step 6 the user can see the current status of the physical process in terms of activation in the model and the feedback can introduce a new cycle of execution if new activation is enabled.

5 CASE STUDY: INDUSTRIAL LIFTER

An industrial lifter example shown in Figure 20 is used for the case study. The lifter is designed to work in a two-level conveyor system. Pallets may circulate in such a system transferred by so-called “Start” and “End” lifters between the levels. In this lifter there is a sledge conveyor, which can move the pallet to any direction and two other conveyor segments which can either upload the product or download it, that means can move in only one direction.

The lifter’s responsibility is to transfer the pallets between the conveyer levels. Thus it provides the pallet circulation between the conveyor levels. To develop the tool this lifer is used as the underlying equipment component. The tool generates a model where all the functional components (conveyor segments) are represented as concepts and the intersections among them are the relationships.

In the Crossword tool developed to model with Connectionistic grids, a lifter model shown in Figure 20 is developed.

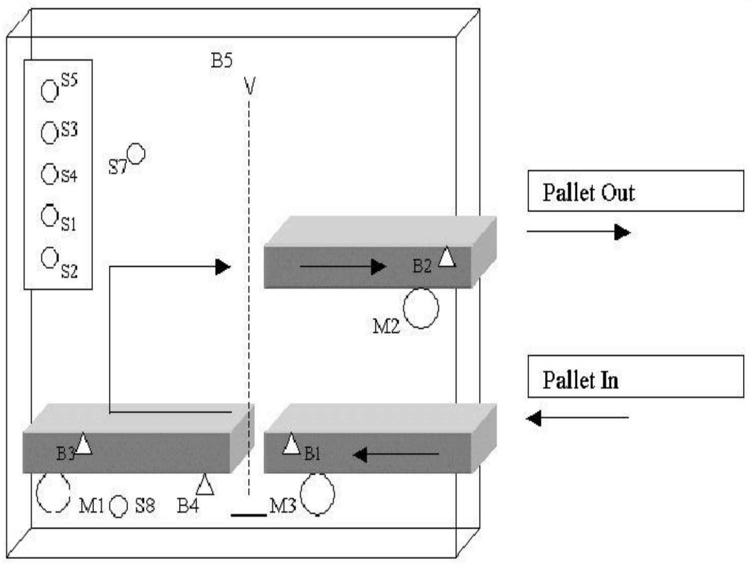


Figure 20: Functional components of the lifter [20]

In the “model” layer of the execution chain, the model of the lifter is developed using the tool. For simplicity of the demonstration, a part of the model is shown in Figure 21. To understand the whole scenario, consider that a pallet has been downloaded to the lower conveyer segment of the lifter and the pallet is to be lifted up to the upper lever conveyer. The user of the tool wants to do this task by the model, developed in the tool, as shown in Figure 21. The implementation of Figure 21 grid in the tool interface is shown in Figure 22.

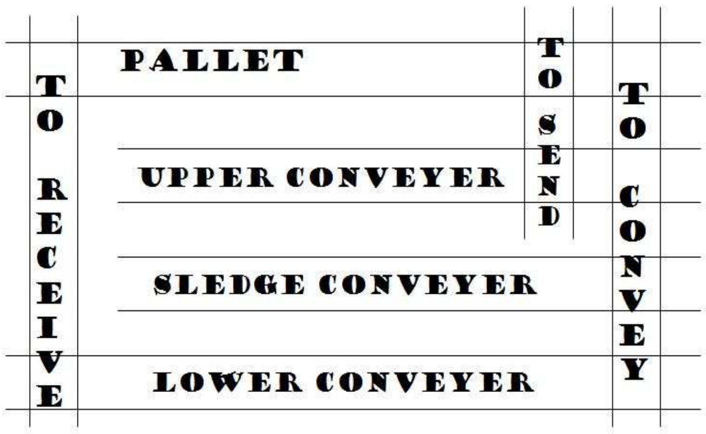
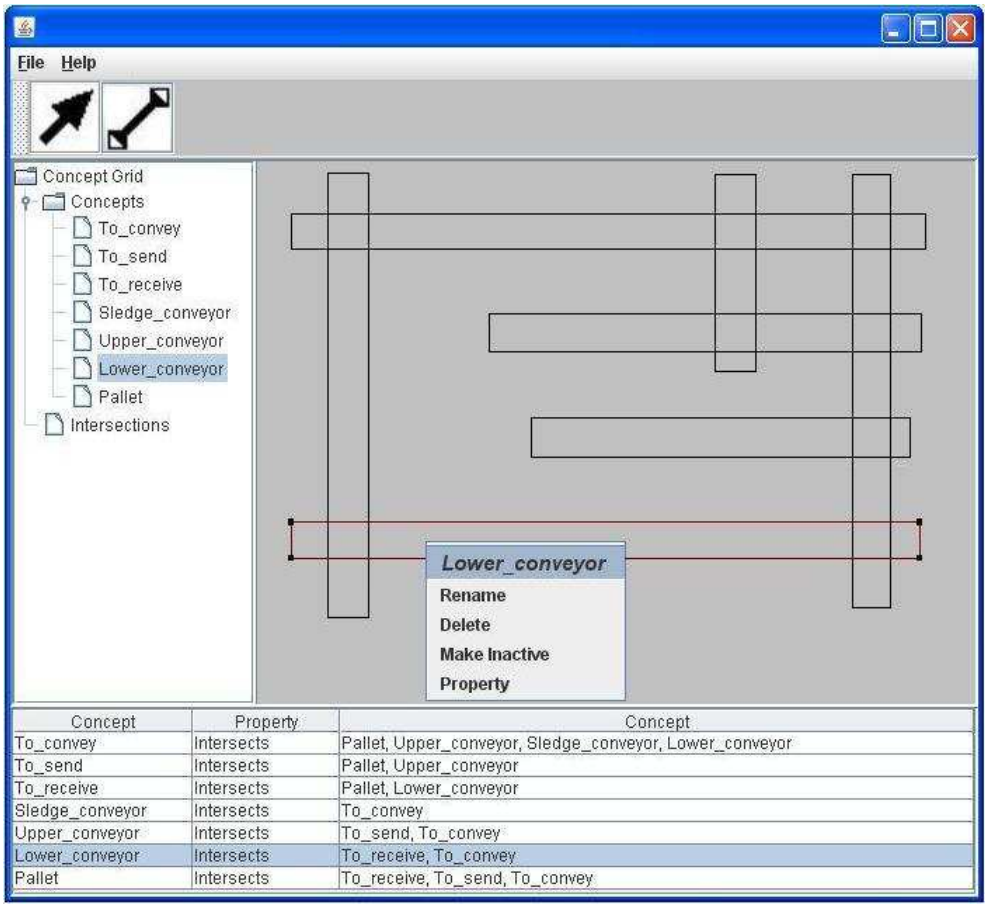


Figure 21: *CCG* model showing the lifter components [29]

Let us denote the “lower terminal” as c1, “upper terminal” as c2, “sledge” – c3, “to send” – c4, “to receive” – c5, “to convey” – c6, “pallet” – c7. Therefore a CoC set (1) as takes the following form {c1, c2, c3, c4, c5, c6, c7}. The lifter should be capable of performing the following processes, which are encoded as the activations: lifter loading process (a1), pallet transportation from the lower terminal to the sledge (a2), vertical motion of the sledge with the pallet (a3), vertical motion of the sledge without a pallet (a4), pallet unload from the sledge to the upper terminal (a5), pallet unload from the upper terminal to the next piece of equipment connected to the lifter (a6). Therefore the set of activations A is looks as follows {a1, a2, a3, a4, a5, a6}. The activations are defined in terms of elements of the power set of CoC. The activations are defined as follows: a1 = {c1, c5, c7}; a2 = {c1, c3, c7, c6}; a3 = {c3, c7}; a4 = {c3}; a5 = {c2, c3, c6, c7}; a6 = {c2, c4, c7}.

The change between the activations is performed through activate (3), and ι and ω functions (4).

The pallet being loaded to the lower segment is denoted as a1. In order to continue the transportation process of a pallet, the activate function, which is responsible to derive new activations through the grid being updated by ω, once the pallet is loaded activates the next combination of the elements of CoC deriving a2.



Concept	Property	Concept
To_convey	Intersects	Pallet, Upper_conveyor, Sledge_conveyor, Lower_conveyor
To_send	Intersects	Pallet, Upper_conveyor
To_receive	Intersects	Pallet, Lower_conveyor
Sledge_conveyor	Intersects	To_convey
Upper_conveyor	Intersects	To_send, To_convey
Lower_conveyor	Intersects	To_receive, To_convey
Pallet	Intersects	To_receive, To_send, To_convey

Figure 22: CCG model shown in the tool's interface [29]

The user of the tool develops the model as shown in Figure 21 in the tool's visual interface. Here all the concepts are drawn at first. They overlap each other where they make an intersection, and then right clicking on the concept the user can make the intersection. The user can activate or deactivate the concept as shown in the Figure 22.

5.1 Evaluation

A knowledge-based approach inspired by the Connectionistic theory was proposed for the modeling and controlling of automated systems. The approach provides with a formal but still flexible structure of knowledge – *CCG*, which may help to overcome current limitations in expression of experts' knowledge. The set of modeling principles and formal notations was developed for implementation of the approach. This approach has not been applied in the domain of factory automation. That is why the question arises that if that approach can be applied to this domain with the aim to solve the limitations of other approaches. This very question is addressed in this research work.

To do that research work, the background of the problem domain is studied along with the solutions provided by other approaches. From the study the conclusion is that the Connectionistic approach has some unique features than that of other approaches. That positive conclusion derived to develop a tool that can implement the initial solution model to analyze more realistically.

The initial prototype of the software tool "Crossword" was outlined to support the present knowledge structure; the software was virtually applied to the case study of pallet-based lifter used in electronic assembly. The initial results demonstrate that the introduced approach can be used to capture engineering knowledge pertaining to manufacturing process. So, the approach could be considered as a potential solution for the implementation of reconfigurable control systems.

5.2 Limitations

The Connectionistic approach which is a natural theory has not been applied in the field of automation system. Therefore the solution model that can be applied to the automation system is not established. Moreover, up to know all the configuration intelligence is based on ontological knowledge-base but in this thesis a whole new approach is proposed which makes the task more challenging. There are still many rooms to improve before using this approach and the tool in a commercial development environment. The tool that is developed here is the first approach to establish this concept. The main research goal here is to develop a suitable solution model derived from Connectionistic theory. That goal has been partially fulfilled within the limited scope of research for a Master’s thesis.

There are other limitations related to experiment environment. The tool is used in the ordinary personal computer environment which has many limitations and differences than that of the computers used in the commercial factory automation systems. They differ in respect of processor and operating system which are very important for computer system. Even though there are differences in the environments where the tool is developed and where that will be used, it provides a solid understanding about the applicability of this approach in the domain of manufacturing systems.

In this thesis Connectionistic theory is used to establish the proposition and the development of a solution model for the automation of software configuration in factory automation. A tool has been developed that shows the prototype implementation of the model. This is the beginning of the new area of research. From the learning of this research is that, it would be worth to continue research for the development of the Connectionistic Grid in this domain.

6 CONCLUSION

Connectionistic approach is a natural science theory. Factory automation system is an application area where this approach can be applied to enhance automation process further. Crossword is the tool support to make that automation enhancement possible. The research question here is to find the applicability of the Connectionistic approach by building a suitable model. The tool is to support the theoretical aspect with a solid implementation to continue further research.

6.1 Summary

The Connectionistic principle is that mental phenomena can be described by interconnected networks of simple and often uniform units. The form of the connections and the units can vary from model to model. In the Connectionistic grid model, it has been simplifies to a concept comprises of arcs with names and properties and the intersection of those arcs. In this grid model they are called *Concept* and *Intersection* respectively. The concepts can intersect each other which form the interrelationship between concepts. Concepts and intersection all together form the grid.

The grid is used to represent knowledge of human engineer that can be used to reconfigure the mechanical devices used in factory automation system. The idea is to develop such model to represent human knowledge. The tool Crossword is to provide a GUI for the user to build Connectionistic grid to configure the system without having specialized knowledge. The whole idea is to provide the users of the tool with relatively less knowledge of how the software changes are able to reconfigure the software in the machine in production systems. To develop such a system and to establish the theory of Connectionistic grid in the field of factory automation is the main goal of this thesis work. Keeping that aim in mind, the background has been studied and limitations of other solutions have been identified; the applicability of the Connectionistic approach has been studied and a suitable concept model has been established. A tool has been developed to verify the model in reality. It has been applied for a case study. After that a logical argument has been given for the evaluation of the research within the scope.

6.2 Future Work

The use of a relatively new theory in a new field is a massive task that requires a lot of research and development. The model needs to be established from the theory to apply it in the real application systems. The software tool developed for this purpose is a starting point to shape the theory and its use in the domain of factory automation. The tool has been built from scratch so there is much room to improve and add many feature based on the changes in the model.

This research addressed the knowledge representation but did not address the reasoning of the model. Therefore a subset of the solution domain has been studied here. To complete the picture the reasoning part is a must which leave room for future research.

In order to adapt Connectionistic approach in the field of manufacturing control, further research need to be carried out on the complete solution domain. The principles of knowledge reasoning in CCG have to be further clarified and structured. Further study on reasoning may require some change in the current knowledge representation model, CCG. The "Crossword" tool should be further developed and tested with more elaborated cases including the test-bed of a production line for electronics assembly; the usability of the tool to be explored more intensively. The performance of the algorithm needs to be optimized at some point of time for large CCG, because the performance of the algorithm is very important for any real time system.

REFERENCES

[1] R. J. Brachman, and H. J. Levesque, Knowledge Representation and Reasoning, *Morgan Kaufmann, 2004.*

[2] I. M. Delamer, Event Based middleware for Reconfigurable Manufacturing Systems: A Semantic Web Services Approach. *Doctoral Thesis, Tampere University of Technology, Finland, 2006.*

[3] D. Deneux, and X. H. Wang, "A knowledge model for functional re-design", *Engineering Applications of Artificial Intelligence, 2000, 13, pp. 85-98.*

[4] National Institute of Standards and Technology (NIST), an analysis of existing ontological systems for applications in manufacturing and healthcare. *Internal report no. 6301, 1999.*

[5] A. Gomez-Perez, M. Fernandez-Lopez, O. Corcho, Ontological Engineering: With Examples from the Areas of Knowledge management, E-Commerce and Semantic Web. *Springer-Verlag, London, 2004.*

[6] N. Goldblum, The Brain-Shaped Mind. *Cambridge University Press, 2001.*

[7] A. M. Meystel, and J. S. Albus. Intelligent Systems: Architecture, Design, and Control. Wiley Series on Intelligent Systems, Albus, J. S., Meystel, A. M., and Zadeh, L. A. (Eds.), *Wiley-Interscience, 2000.*

[8] L. Shastri. "Advances in SHRUTI - A Neurally Motivated Model of Relational Knowledge Representation and Rapid Inference Using Temporal Synchrony", *Applied Intelligence, 1999, 11 (1), pp. 79-108.*

[9] A. Collins, and E.F. Loftus, "A spreading activation theory of semantic processing", *Psychological Review. 1075, 82, pp. 407-428.*

[10] Z. Hu, E. Kruse, and L. Draws, "Intelligent Binding in the Engineering of Automation Systems using Ontology and Web Services", *IEEE Transactions on Systems, Man, and Cybernetics - Part C: Applications and Reviews, 2003, 33(3), pp. 403-412.*

[11] B. Kulvatunyou, and N. Ivezic, "Semantic Web for Manufacturing Web Services", *In processing of 5th Biannual World Automation Congress, 2002 June 9-13, Orlando, FL, USA, 14, pp. 597-606.*

[12] M. Obitko, and V. Marik, “Ontologies for multi-agent systems in manufacturing domain”, Proceedings of the 13th international workshop on database and expert systems applications – DEXA’02, *Aix-en-Provence, France, 2002 September 2-6, pp. 597-602.*

[13] E. L. Rissland, “Artificial Intelligence: knowledge representation”, Ch. 4 in Cognitive science: an introduction, second printing, Stillings, N. (Ed.). *Massachusetts Institute of Technology, 1987.*

[14] E. Rosch, and C. B. Mervis, “Family resemblances: studies in the internal structure of categories”, *Cognitive Psychology, 1975, 7, pp. 573-605.*

[15] C. Schlenoff, M. Gruninger, F. Tissot, J. Valois, J. Lubell, and J. Lee, “The Process Specification Language (PSL): Overview and Version 1.0 Specification”, *NISTIR 6459. National Institute of Standards and Technology, Gaithersburg, MD, 2000.*

[16] C. Schlenoff, R. Ivester, D. Libes, P. Denno, and S. Szykman, “An analysis of existing ontological systems for applications in manufacturing and healthcare”, *NISTIR 6301, National Institute of Standards and Technology, Gaithersburg, MD, 1999.*

[17] W. Bechtel, and A. Abrahamsen. Connectionism and the mind: an introduction to parallel processing in networks. *Cambridge, MA: Basil Blackwell, 1991.*

[18] D. Rumelhart, and J. McClelland, Parallel distributed processing: explorations in the microstructure of cognition. *Cambridge, MA: MIT Press, 1986.*

[19] R. W. Lewis, Programming industrial control systems using IEC 1131-3. *The Institution of Electrical Engineers, ISBN 0-75296-950-3, London, 1998.*

[20] A. Lobov, “An approach the formal verification of automated manufacturing systems with programmable control”, *M.Sc. Thesis, Tampere University of Technology, 2004.*

[21] SOA4D, DPWS implementation, https://forge.soa4d.org/, [*Accessed: November 2008*].

[22] International Electrotechnical Commission, IEC TC65/WG6, “Committee Draft: Function Blocks for Industrial-Process Measurement and Control Systems - Part 1 Architecture”, *International Electrotechnical Commission, 2003.*

[23] W. Bechtel, and A. Abrahamsen, Connectionism and the mind: an introdiuction to parallel processing in networks. *Cambridge, MA: Basil Blackwell, 1991.*

[24] J. L. Martinez Lastra, Reference mechatronic architecture for actor-based assembly systems. *TTY-Paino, Tampere, 2004.*

[25] M. G Mehrabi, A.G Ulsoy, Y. Koren, Reconfigurable manufacturing systems: Key to future manufacturing. *Journal of Intelligent Manufacturing, Vol 11, No 4,*

August 2000, pp 403-419.

[26] T. Gruber (1993). "A translation approach to portable ontology specifications". *In: Knowledge Acquisition. 5: 199-220.*

[27] F. Arvidsson and A. Flycht-Eriksson. Ontologies I. Retrieved 26 November 2008.

[28] Ian Horrocks and Peter F. Patel-Schneider. KR and Reasoning on the Semantic Web: OWL. *In Handbook of Semantic Web Technologies, chapter 9. Springer, 2010.*

[29] A. Dvoryanchikova, Md. M. Hossain, A. Lobov, J. L. Martinez Lastra, & I. Hammouda, 2008. A connectionistic knowledge-based approach to the modelling and control of manufacturing systems. *Proceedings of the Lighthouse of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, I.T. Revolutions 2008, December 17-19, 2008, Venice, Italy 7 p.*